B

Personal Viewers
Information at your Fingertips. . .Literally

Note: This memo is a partial draft. The material through Section 6.5 is complete and quite stable. It is final enough to comment on and to be useful to think about for anyone currently engaged in new UI design based on web UI principles. While the UI discussed here centers around a new handheld platform that I describe, it and the scenarios apply equally well to the desktop and I intend it as a proposal for a new desktop UI. A later section will discuss this aspect in detail but a lot of it should be pretty obvious. For those needing just an overview, a good intro to the material would be to read Sections 1, 2, 5, and 6 through 6.1.2. The TOC provides a topic outline of the rest.

1 Introduction

A confluence of emerging display, network, and software technologies is making possible a new category of device: a truly useful, portable, personal viewer. This device will be the first handheld computing device that has the necessary attributes to achieve a truly mass-market penetration. It has the potential to become a necessity as well entrenched and ubiquitous as the telephone. Everyone will have at least one, and usually more than one. It will not only be useful as a device unto itself, but also as a complement in the user's desktop PC setting and in meetings.

The opportunity for defining this platform and providing software and content for it is, obviously, large. Worldwide, ultimately *billions* of devices. There is opportunity not only for the operating software royalties, but also for systems, content, and merchant services that will support these devices.

The basic idea of handheld computing devices is hardly new. We have electronic organizers, Win CE devices, and primitive electronic "books". None of these have yet gained wide acceptance, and are unlikely to, at least to the degree I'm talking here. But a new embodiment of some of these existing ideas, taking advantage of emerging technologies and with a different focus on design and application, especially UI design and new application scenarios, can change all that.

If I'm right, we have an opportunity to define a new hardware reference platform and some protocols and applications based on an evolution of Windows CE, and in so doing create a platform whose volume numbers will dwarf anything we've seen with Windows so far. At the same time, the applications and UI for this personal viewer platform will complement and feed back to the desktop version of Windows. Desktop UI will be simplified, individual and groupware scenarios facilitated, and synergies created that strengthen our lock on the desktop.

Eventually, as these devices evolve, I believe a gigantic role reversal will occur. These devices will become the *dominant computing platform*, taking over from the desktop. Just larger computers evolved into peripherals for a new, dominant platform—the PC—I believe that by 2010 we'll see PC's evolve into peripherals for the new handheld devices.

Even if you don't buy into anything I say about personal viewers as a device, or you think I'm too early from a hardware point of view, I think you'll find that almost all the scenarios are important desktop scenarios too—scenarios we don't do a good job of today. The UI, too, is designed to be applicable to the desktop with minor adaptation. It has a lot that can inform desktop UI for achieving simplicity, removing UI clutter, and realizing the web UI metaphor as the basis for the entire UI (not just pieces of it as people seem to currently envision). The UI mechanisms I discuss also heavily leverage important infrastructure components like rich linking and unified storage, as discussed in my prior memos, which is important for keeping applications married to our platform.

Before I launch into the design goals and details... What I lay out here in terms of hardware and software functionality is ambitious. It's meant to define a class of devices that can penetrate in numbers rivaling and ultimately exceeding the telephone. Incremental steps will most likely be needed, with attendant compromises, especially to meet device cost objectives early on. For example, if handheld displays that rival print quality don't materialize soon enough, then PV-like devices that are lower quality can still be built, as can devices with less functionality if necessary to keep size or cost in the right range. All that happens in these cases is that the subset devices have a smaller, though still probably quite large market. And we continue to improve them till we realize the full vision.

That being said, I'm optimistic that most of what I lay out here is achievable in three to five years, with a good subset doable in two years if we work hard.

2 Goals

The goal of the Personal Viewer, or PV, is to drive IAYF to ubiquity, to make information available not only at your desk or in your den, but anywhere, anytime. To do this, it must make reading information as friendly, convenient, and easy on the eyes as reading a book. It must make electronic text the equal of printed text in terms of user acceptance.

Beyond supporting information as a goal, the PV must also let users capture information in rich ways, including text notes and audio dictations, since doing this is central to the viewing activity. While the creation and manipulation of content on a PV is less of a mainline scenario, it must be possible, and work well within the UI framework.

A more detailed statement of the goals is:

- Make the PV a great way to view all kinds of online content (documents, books, newspapers and magazines, web content).

- Make electronic books and periodicals a reality; that is, make these volume applications. This includes making online purchasing and subscription for a broad range of content a reality.
- Make the PV a great tool for note taking, collaboration (shared presentations, documents, and instant messaging), remote access, handheld PIM, and rich information capture.
- Make the PV not just a desktop companion, but an *extension* of the desktop. That is, useful not only away from the desktop PC, but also in the same room, in conjunction with it.
- Drive UI simplification and innovation that makes the above possible and that can feed back to desktop UI.
- Provide a UI that is scalable and customizable enough to work well for a range of devices spanning the lowest-end handheld book reader to high end desktops.
- Drive protocol and interface standards that tie this strongly to Windows desktop platform, and that establish the PV as itself a strong platform that is hard to clone.

The following is a brief summary of the key characteristics the PV needs to meet these goals (in its mostly richly featured version—subset devices will exist). The rest of this memo will cover the PV design and its uses in much more detail.

- The PV is defined as a hardware reference platform for OEMs.
- It has the display quality and physical size/weight/feel characteristics to make it about as good as holding and reading a book.
- It opens like a book to reveal dual displays (cheapest versions may have a single display).
- In most cases, it is a sealed unit with no buttons, connectors, or slots. The battery in the nicest versions is recharged by induction.
- Battery life is long thanks to the use of new passive display technology, very aggressive power management that is tuned to the PV's tasks, and possibly nonvolatile RAM.
- The UI is controlled by touch. Advanced models have enough touch resolution for digital ink and print recognition.
- The UI is page oriented in the extreme. Viewing and navigation are done by touch operations to flip pages and follow links. There is no need for scrolling and the attendant scrolling controls, or for making choices from menus/dialogs.
- Depending on the model, applications include viewing (electronic book), text and voice annotation, document sharing and collaboration, remote access, web browsing, and streaming audio access. (Eventually, image capture and streaming video as well).
- IR and wireless modem links and standard remote client protocols let the PV do document sharing in meeting settings and with the user's own permanent (corporate and/or home) desktop.

- The PV acts as a ZAW client. An additional protocol lets the PV keep focus sync with the user's desktop machine or any other desired machine accessible to the PV via IR or wireless modem.
- Low end models will retail initially around $299, dropping to $99. This includes a B&W single display, touch sensitivity sufficient for viewing/navigation, and an IR link. This version does not include ZAW client or remote access capabilities; it is used mostly as a standalone device with desktop sync, like today's PDAs.
- The highest end models will retail initially for about the cost of a laptop PC, dropping eventually to $500. These include dual color displays, pen input, wireless modem, voice recording, streaming audio playback, and full ZAW/remote client capability.
- Options will include IR-linked keyboards, headphones, and DVD-ROM/RAM drives.

3 Why Computer Displays and Handhelds Suck

If there is a successful handheld device that has widespread appeal and dominance, it is the book. So it's natural to think, wouldn't everyone want a device that could hold any book of any length, even several books, and give you all the nice features that a computerized book reader could? Not to mention all the applications that go beyond simple book reading.

Well, no, people don't want that, and for a simple reason. It's not so much that the size, weight, or cost of such a device would be unsatisfactory—technology can solve those problems (more on this later) but it wouldn't be enough. And it's not because people wouldn't appreciate nice features like searching, hyperlinking, multimedia content, and so on.

No, the main reason is that reading on computer screens sucks. It sucks so much we'd rather print out a 20 page document (let alone a 200 page one!) than read it on screen. And anything that sucks this much on a big screen is sure to suck worse on a small one.

There are several reasons for this that I elaborate below. For all their variety, bear in mind that they all come down to basically two things: eyestrain and physical fatigue.

- Flicker and other display artifacts. Even on high resolution, high refresh screens, readability is compromised by artifacts like flicker, imperfect geometry and focus, optical distortions and reflections due to the glass envelope, and the active lighting of the surface. By contrast, a printed page is rock steady, geometrically perfect, higher resolution and contrast, and passively lit. The result is much less eye strain.

- Poor presentation for reading. Displays, as currently used, try to pack a lot of information on the screen. Margins are minimal, UI clutter is high, and information is usually displayed in a single, very wide column. This is hard to read because the eyes have to traverse vast distances compared to the narrow columns of a book or magazine. Narrow-column presentations are no favor here because of the scrolling problem. Giant 17" and larger screens are no favor either. Even the head starts to get involved in moving the gaze, a bad thing.
- Scrolling. Display screens, as currently used, are a scrolling medium. This means that the eye has to track text as it moves vertically. It is much easier on the eyes when text doesn't move, and when information scrolling is done by page turning, as in a book. The eye is much happier to scan horizontally than to track vertically.
- Small window. Despite their large size, display screens usually provide a limited window on the information, less than a printed page. By contrast, books and magazines give you two full pages at a time, which helps greatly in keeping a contextual sense of the reading matter. This also makes skimming, a crucial reading activity, much easier. Large pages also give you more space for jotting margin notes within reasonable distance of the paragraph you're notating.
- Physical restriction. Display screens force you to sit in one place, in pretty much one position. This is even true of laptop computers and to a fair degree, today's PDAs too (in the case of PDAs, because the UI is hard to use in a true handheld manner—you have to put them on the table). By comparison, with printed matter, you can sit anywhere and in any position you like. You can shift around, lean back, sit forward, change posture, adjust viewing angle and distance. You can pick the book or document up, lay it or prop it on the table, make any number of almost microscopic changes to how you sit and hold it. The result is less fatigue and greater comfort. To sit in front of a display screen is like being told to sleep all night lying on your stomach and not being allowed to move. The result is uncomfortable and fatiguing rather than restful.
- Inconvenient annotation. The fact is, it's much easier to annotate printed matter. It's true that annotations typed into a computer are ultimately more useful because the text can be searched and shared. But compared to jotting annotations with a pen, the process of entering a text annotation is more time consuming (for the typical, short annotation anyway), more disruptive of the reading process, and also less flexible (you can enter text, but not so easily enter diagrams or otherwise mark up the printed text).

Today's Win CE PDAs are no answer to most of these shortcomings. Display contrast, viewing angle, and resolution are much too low. The "small window" problem is even worse than on a desktop display. Battery life is too limited, and the susceptibility to damage is too great (you can't just toss a PDA around like a book, pile things on top of it, spill coffee on it, etc.).

Besides physical problems of current PDAs, the UI is much too complex and cumbersome if the primary task is reading. You don't just open the thing up and start reading as for a book. And to do UI manipulations of almost any kind, you have to put the device down, so you can't use it like a book anyway. Not to mention that half of the thing is a keyboard that gets in the way if all you want to do is read.

Overcoming all these problems with an electronic device is clearly a tall order. But with a combination of emerging display technologies, some highly integrated chips, and good software, I believe the time is right for us to attempt that task.

4 Personal Viewer Hardware Description

In this section I'll give an overview of the PV hardware platform, including different flavors of it. In the next section I'll discuss scenarios for its use.

Note that my intent here isn't to write a spec for the PV hardware platform or to cover all it's characteristics. I mainly want to hit the salient points that distinguish it from other kinds of handhelds. A PV project will want to do a detailed enough hardware spec to serve as a reference platform design for OEMs. My focus in this memo is much more on the software than the hardware.

From what I've said so far, the PV must sound pretty much like and advanced PDA. It is, basically, but with two key differences:

- There is an overwhelming focus on making the PV compete with printed text as a way to read information; to make it good enough to be a broadly accepted alternative to printed books and documents.
- Partly because of the first point, the PV's focus is on viewing and capturing information, versus creating or manipulating it. Thus, it is even less like a "mini desktop" than today's Win CE PDAs, and it's UI is much simpler.

Because the PV must be as good as a book, in most incarnations it looks like a book. This means it comes in several standard sizes approximating standard book and magazine form factors (4 x 5 to 8.5 x 11). It weighs about a pound and is half an inch or less thick. It is "bound" like a book, having a flexible spine and has a textured outer surface that is warm and slightly forgiving. It feels nice, more like a book than a device.

In most incarnations there are no slots, buttons, controls, or anything. It is a sealed unit. This allows a mechanical design that makes the PV very rugged, reliable, and resistant to dirt and moisture—and cheaper to produce. The internal battery is recharged by an induction charger and lasts for at least at least 20 hours of typical use, possibly up to 100 hours. (To charge it, you just lay the PV on top of the charger; of course, lesser models may use a physical jack for connecting a charger.) Three factors improve battery life:

- Passive display technology, discussed later.
- Aggressive power management, also discussed later.
- Flexible battery technology, allowing much of the case itself—the supporting structure of the PV—to be made up of battery

The PV opens like a book to reveal two display screens like the pages of a book. The screens are almost the full dimensions of the PV; i.e., there is almost no surrounding border. In fact, for models that use a flexible display surface (an emerging technology I'll cover later), there needn't be a physical border at all, not even at the seam where the PV folds. This would allow a nonbroken display image when the PV is held vertically (one screen over the other).

Because it is small, light, rugged, sealed, and simple, you really can take a PV with you anywhere, toss it around, drop it, pile things on top of it, sit back in a couch with it, take it into the bathroom with you, read it in bed, or prop it in front of you on the breakfast table. You can treat it like a book.

A PV can be held several ways: horizontally like a book for viewing pages side by side, vertically for viewing large format pages that span the two screens, and folded over (displays on front and back) for reading in a more compact form.

Opening the PV turns it on of course. There are no switches or buttons. All you see when you open a PV are the two display screens.

The displays in most models are passive LCDs, using new technology reflective LCD's that are very bright and have high resolution (200-300 dpi), contrast, and viewing angle. When read under the same light as a book, the print looks as good as on a printed page. The use of passive LCDs helps preserve battery life. The display surface has a *very* mildly matted finish. This helps reduce reflections and also minimizes the visibility of fingerprints.

The display surface and the edges of the PV are touch sensitive. This is used for controlling the UI of the PV, and optionally for pen input. Depending on the model, the PV may also include a speaker for audio playback and a microphone for voice input. All PV's will include an IR link. More advanced models will include a wireless modem for cellular, PCS, or satellite links.

A 133 Mhz Pentium processor is adequate for running most models of PV. All the firmware is in flash memory that can be updated via the IR or wireless data link. Because the PV is reading-oriented, it uses a very aggressive form of power management. In most cases, the device sleeps between each page turning event. The power management algorithm adapts to usage, so that if the user starts doing input operations, or is doing rapid page turning, the warm-down delay lengthens accordingly.

Because of its IR and optional wireless modem capability, the PV needn't store an entire book or document in memory at once. As long as it has connectivity to a data source, it can access information incrementally, treating the memory as a cache. It only needs to pull down a complete work when the user will be out of connection range. Advanced PV models exploit connectivity to support streaming media as well as network-based document storage.

While early models of PV will probably use mainly IR to the desktop for wireless communications, within the next 3 years it will be possible to build advanced PV's that include wireless modems for cellular and satellite data services. This is made possible by two trends. First is the availability of single-chip radio transceivers, which reduce the size, power drain, and cost of the wireless capability. The second is the coming availability of low earth orbit satellite data services. These let mobile devices use lower power transmitters and free them from the constraints of directional antenna pointing. The "wired planet" capability this creates will open new scenarios and implementation approaches for devices like the PV, a number of which I cover in this memo.

This ability to transparently tap into a network-resident workspace is key to how the PV will evolve to become a dominant computing platform over the years.

PV's will probably need about 16 MB of RAM, in addition to the flash memory for the firmware. Ideally the RAM will be nonvolatile, too, as an aid to battery life. In principle you could get by with less RAM because you can store quite a bit of text in a much smaller memory space. However, the better PV models need to handle multimedia content, which demands some cache room. Internet browsing also needs enough RAM for running scripts.

The cost of the PV is key to its achieving ubiquity. It needs to be available in models so cheap yet nice to use that people really will be happy to use them in place of a printed document or book. Low-end models need to start in the $199 to $299 range, dropping eventually to $99, if they are to become mass-market devices on the scale I'm envisioning. As a book/document replacement, it's sufficient for a low-end model to have a single, monochrome display, with no pen input, no audio, and an IR link only.

Advanced PV models will support color displays with motion video capability, pen input, wireless modems, and ultimately a built-in imaging chip. The imaging lens would be on the inside of the PV, above one of the display screens, making it useful for videoconferencing. When the PV is folded so that the displays are back to back, the lens would face outward, letting the PV serve as a digital camera. The display screen on the other side from the lens would show the image preview and also the UI (touch buttons) for controlling the camera functions.

Key to realizing the PV, especially at the right cost, is the availability of appropriate display technologies. While no currently produced displays combine the right cost, resolution, contrast, viewing angle, and power consumption, new technologies have been announced that should be able to do this. Section 8 mentions some of these. Best of all, many of these technologies are passive, and at least one promises to realize a flexible display substrate that can be cheaply produced.

Finally a word on accessories. Some users will want to use the PV as a laptop replacement, so an IR linked keyboard/mouse will be available. This will include a detachable bracket for holding and propping up the PV. The keyboard is weighted and molded so that the combination balances and sits well on a table or on the user's lap. The bracket detaches so that the PV can be propped up separately from the keyboard.

DVD-ROM and DVD-RAM drives, also IR-linked, will give the PV access to recorded media, such the full copies of electronic books, video, and other multimedia content. The DVD drives would be useful for the PV models that lack a wireless modem, where the content to be viewed is not made available online by the publisher, and in cases where the user won't have wireless connectivity.

Finally, an IR-linked headphone and headphone/mike combo would provide high fidelity sound playback and playback privacy.

5 Scenarios

In this section I'll present a brief description of daily activities using a PV. This will cover many of its important functions. I'll avoid getting specific about UI particulars here, since I cover those in the next section. But I'll at least say this: the PV UI is very unlike today's windows UI, so don't try to imagine today's UI on this thing.

The PV instead uses a much simpler "web UI" that essentially does a way with windows, scrolling, drop-down menus, and other UI clutter. Yet most common operations can be done with one or two touches on the screen. And in comparison to today's UI, the PV's web UI gives up nothing in terms of feature richness and customizability. To the contrary, it is more customizable and easier to add features to.

Note that in this scenario description I'll assume one of the higher end versions of the PV. This lets me illustrate a fuller range of PV uses.

It's early morning, at home. My PV sits on its charger in the bedroom. I like to read news while I eat breakfast, so I pick it up on the way downstairs and put it in front of me on the kitchen table, opened and propped up in a stand, with the displays vertically oriented (one over the other). This provides a single, full-page presentation (newspaper-like).

The PV knows what time it is and it knows that around this time I like to read the morning news, so when I open it, it turns on and by default displays the front page of my personalized newspaper. (This newspaper was synthesized overnight on a server according to my user profile and stored in my network-based workspace, which the PV accesses.) Using touch operations I can page through the newspaper or follow hyperlinks. Since all information is page formatted, I never need to scroll.

What I see on the screen looks exactly like a newspaper page, albeit on not quite as large a page (my PV is 8.5 x 11 fully opened). Because the screen resolution is very high, even the tone and texture of the paper can be conveyed and the fonts look much better than screen fonts. They are at print resolution, so they look like print fonts.

There are no UI controls, icons, or anything like that; just the newspaper page. I'm able to page back and forth through the paper or riffle through it by touch operations on the left and right edges of the PV. This is an important point—the basic activity of reading involves *no* visible UI. Nothing detracts from the illusion of a printed page—at least, not until I need to do operations that go beyond what you can do with a paper document.

Back to the newspaper, I see some stories I'll want to refer to later during the work day. I use a fingertip or a tap of the pen to "clip" these. I'll be able to review these news clippings later, on either my PV or desktop machine, by viewing a Clippings Folder.

Note that any kind of information in any context can be marked as a clipping, not just items in the newspaper. A clipping is just a link to the "clipped" information (though for transient information an actual copy is captured in the Clippings Folder.)

OK, time to head to the office. I start my day at my desktop by reviewing my personal work newsletter—what I guess we're starting to call the user's start page. (See my Beyond Browsing and Hyperactive Desktop memos for a discussion of the personal newsletter.) I delve from there into my inbox, which has lots of stuff to read. Since my PV is a much friendlier reading device, I pick it up and sit back.

Via its IR link, the PV knows it's in range of my desktop PC. By default it syncs with the desktop's focus, so that when I open the PV up it's showing what my desktop PC is currently showing: my inbox. (There may be other clutter on the desktop but the PV doesn't show this: it just gives me what has the focus, in this case the inbox).

The PV's default presentation of the mail differs from that of today's desktop. This is because the PV provides a more document-like metaphor for viewing information. Instead of special-purpose multipane views, the PV leverages its facilities for presenting information in a multipage format with flexible viewing and navigation operations within and between documents.

So, when I pick up my PV to view my inbox, what I see is a document. This document, the inbox, presents links to the messages its contains, and to the folder hierarchy it's a part of, so it's easy for me to look at what I need to.

Views on the inbox show its contents in all the ways I want to see mail headers, with the full gamut of options for sorting, filtering, categorizing, textual and graphical threading, and inline preview text. And because the folder is a document, I can edit any of these views, dressing them up with annotations, text, and graphics, and even (for certain of the views) spatially rearranging the header items.

To make things even nicer, several of the more common inbox views are preconfigured as "sections" of the inbox document. I can view those by linking to them from the TOC for the inbox document, or jump to them directly via bookmarks that appear as tabs in the margins.

As you'd expect, to read a given message, I tap its link in any of the inbox views. Long message and folder documents spill across sequential pages, so to read through them I just flip pages.

Messages and folders are all arranged in an implicit next/previous order according to the folders' current view, so flipping past the last page of a message for example takes me to the next message in the folder, or else to the next folder in the hierarchy of folders. Likewise, flipping back past the first page of a message takes me to the previous message, or else to the previous folder. I can also use Next/Previous commands to navigate among the sequence of messages and folders, including to move up and down the hierarchy (for example, to return directly from a message to the inbox).

What's significant in all the above is that I'm not faced with a UI that requires multiple panes, scrolling, or special-purpose navigational or viewing conventions. My mail is like a well-organized set of documents and I can read it as such. A more desktop-like two-pane display mode showing a folder list and message content in separate panes is possible on the PV, but not the default.

Selecting messages and performing operations like reply, move, and delete are handled by touch operations. Later sections discuss how these and other functions work at the UI level.

Now, the PV isn't an input oriented device, so I won't typically issue replies from it. The only case where it would make sense to reply from the PV is when my response is so short that a quick handwritten or print-recognized note will do the job. Otherwise, I'd much prefer to compose the response on my desktop, because typing is quicker than writing.

Luckily, this is easy. While viewing a message I want to reply to, I can just put the PV down and return to my desktop keyboard. Because the PV and desktop keep sync on the current focus, the message I want to reply to will be in view on my desktop screen. Or, if I've already opened a reply form on the PV, that form will be open on the desktop. I complete my reply from on the desktop PC and send it off.

The preceding sequence of mail actions shows how the PV can act as an extension to the desktop, not just a companion. Because it's a much friendlier reading device, I can pick it up at any time to read what's on my desktop screen, or go back to the desktop any time I need to do manipulations that aren't convenient to do by touch or with a pen. The devices stay in sync automatically so switching between them is a no-brainer.

Returning to my personal newsletter, I now delve into the section on project updates and come across a long document I need to review. I'll of course read this on the PV, making pen annotations as desired, or, when longer annotations are needed, keyboarding them on my desktop. If my PV is so equipped, I can also make voice annotations. As with the mail example, I can switch back and forth between the devices at any time, and the correct context will be presented.

The PV makes it easy for me to do rich note taking on documents. I've already mentioned how a single tap operation lets me "clip" material from a document for easy review later. I can also use my fingertip or the pen to highlight stuff on the page, with selectable highlighting colors. Highlighting shortcuts make it easy to highlight whole regions; for example, I can highlight an enclosed shape, and by holding the highlighting tip still at the end of the closed figure, the region fills with highlighting. With the pen I can jot notes anywhere on the page, or insert embedded notes (annotations). Using the pen or the fingertip, I can also selectively erase highlights, jots, or annotations.

The nice thing about note taking on an intelligent device is that it can make it easier for me to review notes I've made previously. One way the PV does this is by maintaining a Notes Index for each document that has annotations. The index includes each annotation's title (if entered) or else the first few words of its text, plus the page number. To review a given annotation, I can link directly to it (tap the title), or to the document page that contains it (tap the page number).

While I'm going over the document I get a popup notification of a meeting I have to go to. This shows up on my desktop and PV screens so it doesn't matter which I'm looking at when the popup occurs.

So I take my PV and go.

Everyone at the meeting has a PV. The meeting presenter has called up a presentation on his or her PV, and this is also being displayed on a big screen thanks to the IR link from the presenter's PV. (Or the presenter may be using a traditional PC and projector for his or her presentation.)

Because my PV knows I'm in a meeting now (based on my schedule info) and also recognizes another machine present that is showing a presentation, it by default syncs to it. So, as soon as I enter the room the presentation appears on my screen. (This kind of automatic behavior, which I call IntelliView, can be customized by a preference setting; see Sections 6.2.2 and 6.3.4.2 for more on IntelliView.)

There are several things I want to be able to do during this meeting. The most basic is to keep synchronized on the presenter's focus. Another is to freely page back and forth through the presentation—without affecting the shared display of course! Another is to take notes, either on a separate sheet or directly on my copy of the presentation slides themselves.

If this is a collaborative meeting, I'll also need to be able to make annotations that do appear on the shared display; that is, annotate as well as manipulate the shared workspace. If this is a controlled meeting, I'll need to be able to ask for and be granted "the floor" before I can muck on the shared workspace.

Finally, I'll want to be able to exchange private, instant messages with other people in the room, or to engage in short, private chats with any of them.

Much of this is stuff you can do in Netmeeting, and as you'd expect it all makes sense for face to face meetings as well. The UI details for the PV differ, though, because the emphasis is more on viewing than input, and input is touch and pen oriented. Section 6.9 describes how the above collaborations functions work on the PV. Consistent with its design philosophy, it all involves minimal UI. In fact, a single tap operation is all that's needed to initiate any of the functions I just mentioned. That's important, because it means using the device doesn't distract me from the meeting.

One other thing I can do in the meeting if my PV is so equipped: I can record part or all of the meeting. It's ok if it's a long meeting; the PV is just a cache, and can stream stuff over its wireless link to my server-based storage. It is a wireless, ZAW client.

Naturally, it would be nice for server storage reasons to avoid a dozen people recording the same meeting. This can be solved by having one person, like the presenter, record the meeting in the meeting workspace. Everyone else can then access the shared recording, which would be persisted on the intranet for however long makes sense.

Having an audio recording, even of a long meeting, is actually quite useful. For one thing, if I'm using headphones with my PV, I can skip back and replay something that I missed, right there in the meeting.

Also, as I've previously discussed in my New Technology Areas memo from last year, multiple media streams can be cross-referenced to time as a way to help retrieval after the fact. The result is what I called "rich information capture" in the earlier memo.

For example, as the meeting progresses, a computer record is kept of the current document focus/position on the shared display and on my private display. Every annotation or note I enter is timestamped. All voice annotations I make are also referenced to the realtime timeline.

Later, from my PV or my desktop, I can do interesting things. I can be flipping through the presentation and ask to hear what was being said when I was viewing that slide or page of a document, or when I was scribbling this note or applying this highlight. I can be listening to the meeting recording and watch as the display automatically sequences through the stuff that was presented, in a virtual replay of the meeting. I can search on keywords in the presentation or in my notes, and then listen to what was being said in the context of the found keywords.

As for other PV operations, UI is minimal, and most of what I just said can be done with single tap operations. For example, while playing back a recording, tap on a jot or highlighted area to hear the part of the recording that was made when the jot or highlight was being applied.

This idea of rich information capture and retrieval will ultimately encompass photo and video media as well. Consider a very advanced PV that has a built-in imaging chip. This would give the PV image and video capture capabilities in addition to audio and pen. This may not be that useful in the typical office scenario, but consider factory and warehouse scenarios, where an inspector captures images on his or her walkthrough and annotates them with pen notes or voice recordings. Or, imagine a car rental agent snapshotting a rental car and recording comments about its condition when handing over the keys or getting it back.

Back to my meeting...it's finally over. Unfortunately, I've now got another meeting, offsite. As I travel there in the shuttle, I catch up on my email and other reading. My PV acts as a remote client, so I can do my mail, PIM, and other desktop work from anywhere. (If I need to do much input from remote, however, I'll want to use the wireless keyboard.)

The PV is also an internet client, so I can browse the web. While reviewing my notes from the meeting I just left I follow a link that was on one of the presentation slides. The PV makes exploring the web easier than today's browser, because its browsing context is a graph, not a linear chain.

This means I can explore a bunch of links, back up, take another branch, back up, and continue back down the original path forward—or any of the paths forward from that point that I've been exploring. The UI can intelligently default the path forward when there's a branch point (e.g., follow the most recently traveled branch), as well as offer me the choice, textually and graphically. I no longer need to keep these branching exploration paths in my head, nor do I ever need to manually retrace a sequence of links forward.

That evening at home, I'd like to do some pleasure reading. Using the PV I navigate to Amazon.com and purchase the electronic edition of a book. My PV identifies me to Amazon automatically so I don't need to fill out order forms; I just select the book(s) I want and press the Purchase button. Under the hoods, this gives me a cryptographic certificate that lets me access the book.

Depending on how the content supplier has set things up, a full copy of the book may now be downloaded to my workspace or only chunks may come down as a way to pre-load my cache. Note that by "workspace", I don't mean my PV in the typical case. The PV is a client that accesses my permanent workspace, but that workspace is stored elsewhere—on a server or on some desktop machine. (Of course, things can also be set up so that the book is downloaded to the PV, as there will be cases of PV's that are designed to function as fully self-contained devices.)

As soon as I purchase the book, the right-side display of my PV shows the book's cover. With a touch operation on the right edge of the PV I can now open the book (flip over its cover) to reveal the title page and TOC; or I can dismiss the book ("Back" operation), allowing me to continue ordering through Amazon.com. The book will exist as an item (link) in My Documents, so I can access it at any time in the future.

Let's say I only wanted one book, so when it's cover appears I turn the cover page and start reading. As for my morning newspaper, there's no UI to the book. What I see is just the book's pages, rendered exactly as the printed version would appear. The rendering even includes the book's margins, and a visualization of the stacked pages on the left and right sides; this gives me a visual cue for how thick the book is, and how far into it I am. I can place bookmarks and go to bookmarks with a single tap operation in the top margin, where bookmark tabs appear. Of course I can also enter jots, highlights, annotations, and recordings into the book—a much easier thing to contemplate doing to an expensive book when such things are erasable, or can be temporarily turned off, as you can do on a PV.

Bookmarks and the visual enhancements I just mentioned aren't unique to books of course—I just saved mentioning them until now, but the same applies for viewing any documents.

The PV has audio playback, so how about an audio book or a CD? I can buy those too. It works the same way as the print book example: on completing my purchase, I see the item's cover page (cover art). Below that I'll also get a set of playback controls. (Might be good at this point to put on the IR headphones for decent sound.)

The cover page of an audio book or CD is there for more than decoration. It's the cover for the liner booklet. I can turn the page and read this the same as for a book. In fact, if I'm a fan of this author or music group I'll especially want to, because the TOC for the liner booklet will include links to relevant web sites, like the official site for this book or CD, the author or music group, and the publisher. There will also be links to fan and discussion forums. (Print books will include these links too of course).

When I'm done for the day, I lay the PV on top of its charger and go to bed. Been a fun but long day.

6 User Interface

Two words summarize the UI principles for the PV: minimal and page-oriented. Actually, one will do: book-like.

To restate one of the requirements, the PV content must be as easy to read as the printed equivalent. Besides the physical requirements of size, weight, and display quality, this brings with it UI requirements—requirements that the UI be "minimally invasive" and much like reading a printed document. I believe it must do this or else people won't accept it as a good enough replacement for print, no matter how nice the frills are.

This statement concerns more than the feature set for viewing information. With a print book, magazine, or document it's very easy to capture important information in context by jotting notes, highlighting text, and even clipping material for later reference. The PV must support these activities with equal ease.

Of course, as Section 5 showed, the PV has many uses beyond simple reading and note taking. But reading first, and note taking second, have to be the tasks the UI is optimized for, the UI for the other functions must grow from those roots.

So what do "book-like" and "minimally invasive" mean in this context? They mean that you must be able to pick up a PV and read your book or document without ever putting the PV down (unless you choose to), and without ever mucking with menus, dialogs, or other obtrusive UI. It means that with a very simple action of a fingertip, you must be able to turn pages in either direction, riffle them fast or slow, place bookmarks, return to bookmarks, highlight text and erase unwanted highlighting. This and much more can be accomplished with no visible UI (in fact, everything can be done with a finger except for jotting ink and editing text).

All other functions are icing on the cake and while they too should be minimalist in operation, it will be OK for them to involve a little more in terms of user action or visible UI. The UI for the reading and note taking activities must however be invisible and operable essentially without thinking. It turns out this is possible. A totally naïve user can open a PV, start reading, and if all they discover is how to turn pages (touch upper left and right corners of case) they can navigate to any content in the PV, and learn all about how to operate it.

Now, while my goal is to propose a UI that's great in the context of the PV device and its scenarios, I also do *not* wish to suggest a design departure that's relevant only to the PV. To the contrary, the UI I'm proposing here represents something I believe applies equally well to the desktop, with small adaptations that I will discuss.

Most significantly, the UI is based on a web UI. The PV UI bites off a web UI architecture and, I believe, solves the problems inherent that kind of architecture in innovative ways. The result is a UI that is, at base, scroll-less, windowless, and almost completely widgetless. It is even desktop-less (an idea that is not as crazy as it sounds).

In other words, it is a UI that has essentially zero clutter, and few concepts to master. In this way it certainly is a major departure from today's desktop UI. The user focuses entirely on content and not on UI appurtenances. And it's really the case that a zoo of appurtenances and constructs doesn't exist (versus saying all the gunk is still there but just auto-hidden or recast into analogous forms, like frames instead of windows).

Unlike other web UI designs being considered at the company, this one makes no distinction at all between web UI pages and content pages. "UI" and "content" are the same thing, and exist in the same navigation space. As I'll show, smart next/previous logic and intelligent management of the navigation chain solves the problems that other groups have had in considering this alternative.

Because there's no seam between UI and content, no notion of "dual" spaces, the web UI I'm proposing here is conceptually simpler for the user than a model that has separate UI and content webs. The one-space model is also more powerful and customizable, as I'll explain.

Numerous UI shortcuts and direct manipulations exist as a configurable layer on top of the web UI, so more experienced users can do the most common operations in context, without navigating to UI pages. The user has the best of terse command access plus the richness of the full browser and answer system for exploring the command set.

I'll now go through the major PV functions and discuss their UI. I'll start with a more detailed discussion of the principles of the web UI, then give an overview of the PV specifics. Next I'll through the PV UI task by task and fully describe the UI elements that are used in performing each one. The focus in the latter sections will be on the direct manipulations and shortcuts a user familiar with the PV would use; users less familiar would perform commands by looking then up via help or finding them in the operating manual.

The thing to keep in mind as you read the user activities material is that the web UI is actually the fundamental UI mechanism, and all the shortcuts are things that exist on top of it as frosting. Anything you can do by a shortcut action you can also do by invoking a command found (and explained) on a page somewhere.

A few meta-comments: My intent here is to present a comprehensive vision in enough detail to demonstrate that it is doable, and that enough of the hard issues have at least one reasonable solution. Therefore, the following sections are detailed enough to constitute an initial spec. I've also laid out more functionality than we'd do in any one release, because I also want to show that the design basis is robust enough to cover a broad range of functionality (i.e., it scales).

While some of you may be put off by all the detail, I feel it's necessary to avoid handwaving—the worse of the two evils when the subject of a proposal represents a major change from current design direction. The specific design decisions must of course be subject to review input, prototyping, and user testing so I hope you won't get hung up on details you don't like. What's most important are the requirements, scenarios, and features that comprise the vision, and the proof-of-concept that at having at least one detailed design provides.

6.1 Elements of the UI

6.1.1 Web UI Concepts

The PV UI uses a web architecture, with UI shortcuts layered on top. A naïve user will typically start by using the web UI without the shortcuts—that is, by using the online manual to access UI functions. An advanced user will be able to do all common operations via the shortcuts and without resorting to the web UI.

In this section I'll outline the concepts of the web UI architecture. This part of the discussion is somewhat abstract and applies to any kind of computing device, not just the PV, so I won't even be mentioning the PV here. If you're someone who's not too concerned with the architecture of web UIs, you can skip to the next section, which covers PV specifics, and return here for clarification later.

The basic principle of the web UI is that everything you see and interact with is a document: all content and all UI. These documents are all HTML. There is absolutely no difference between content and UI kinds of documents. In fact the same document can mix content and UI. Even though some UI elements like context menus and toolbars may be presented in ways that don't look document-like, they are in fact implemented as documents and can be manipulated as such.

Please note: other groups like Netdocs are also developing web UIs. While all web UIs will have architectural similarities, the one I'm describing here has some important differences, and I believe major advantages, compared to the other efforts. The architectural definitions I describe here don't necessarily apply to the other designs.

The uniform treatment of content and "UI" pages is important for several reasons:

- You only need deal with one set of navigation controls and conventions, and only one navigation space. You never need think about whether you're in content space or UI space.
- You can use the full power of the UI to manipulate the UI itself. For example, you can annotate, customize and edit UI pages the same as any content (subject to permissions); you can select from multiple views as provided for the page being viewed.
- The model naturally accommodates dynamic content, including downloaded content, that mixes content and UI on the same page or as part of a network of related pages.

To accomplish its "everything is an HTML document" metaphor, the PV greatly leverages integrated storage, as described in my previous memos. The design takes full advantage of storage polymorphism, properties, rich linking, queries, and views. The design could be harder to implement without an integrated store like this, though it would still be possible.

All actions in a web UI occur by clicking on commands that exist on pages. To the user, a command looks like a link, and in fact it is a link. Specifically, a command is a link whose source anchor is the command hotspot, whose destination anchor is the script or code that implements the command, and whose properties are the command parameters. Some commands may run with canned parameters while others may present you with a form for entering the parameters (for example, the Properties command).

Please note that when I say "link", I mean links as I've defined them in my previous Hyperactive Desktop and Storage Unification memos. These are links with rich support for properties and behaviors, many of which were defined in those memos and which I leverage here.

Because a command is a link, it has all the functionality of links. For example, commands can visualize in multiple ways, such as an underlined blue label, a button, an icon, or a graphic image. They can be copied from one place to another, moved, and deleted. Their properties can be manipulated, such as to change their appearance or to preset some or all of their parameters. Everything you can do to a link, or to objects in general (since a link is an object), you can do to commands. This makes rich customizations easy to accomplish, as I'll explain later.

You get at commands by navigating to a page where the desired command is found. The web UI is organized to make frequently used commands a single navigation step away, or through customizations, no steps away. Less commonly used commands may take more steps to get to.

The entire web UI comprises a self-explanatory document, the Operating Manual. This is literally a readable and printable manual that you can go through in a logical order, a page at a time, like any current user manual. The difference is that each command mention is an active command instance that can be invoked in place. A variety of quick ref sheets and indices make it easy to get quick access to sets of commands that are commonly used together, that are logically related, or that are typically used as part of a given scenario. You can also use search to find commands; this calls into play the user assistant when appropriate.

A problem with treating content and UI as part of the same navigation space is that the user's navigation chain gets cluttered with UI-related pages. The web UI solves that through intelligent management of the navigation context, and by making that context a true network, not just a linear chain.

Specifically, when you navigate from one place to another, a new branch in the chain is started. So, if your context is currently B is A-B-C and you navigate to D, then the new context is D in

```
A-B-C
  |
  D
```

Here, D might be a UI page you navigated to from document B. What happens when you click a command on D? The command executes and removes D from the navigation context. Thus, after finding and executing the command, your navigation context is restored.

Now, to find a needed command you might have to navigate along a chain from D to several other pages in the operating manual, resulting in a chain of several steps branching off from B. When you finally pick a command, how does it know what to act on, and what to remove from the navigation context? The answer is: commands operate on the current selection, and (generally) remove the navigation nodes that lie on the branch leading from the current selection to the command (the actual heuristic is a bit more than that for pathological cases, but that's the gist).

Current selections in the web UI are similar to those in today's desktop UI, but there are differences because the web UI deals with a network of active documents that are not quite the same thing as a desktop of active windows. They're similar in that every document can have a single, possibly disjoint selected area. They differ in that you can't use the idea of the current focus to decide what selection a command should operate on. In the desktop world, the document you want to operate on always has the current focus, and all UI widgets implicitly reference this focus. In the web UI world, you may have navigated several hops away from the document you want to operate on, as you looked for the command you need, so focus doesn't disambiguate anything.

Therefore, instead of the current selection being the one whose document has the focus, the current selection is the *most recently selected* area. All commands that are configured to operate on selections will operate on that most recently selected area. Having executed, they'll the trim the navigation context at the branch point that leads from the current selection to the command itself.

The upshot is that you're free to link into the web of UI pages, exploring them as necessary to find the command you need, and then to invoke it. The act of doing so will end up trimming all the UI navigation from the context, leaving you back where you started. Note that depending on the length and content of the navigational path between the command and recent selection, the UI may show you the target and ask to confirm before proceeding.

The algorithm for trimming the navigational context is a bit more involved than what I just said. I don't want to get into all the cases in this discussion, except to say that the effect is to always isolate and remove the branch whose purpose was to find the command that was just executed. Also note that trimming the context doesn't happen in 100% of the cases. For example, an Apply command for property setting leaves the property form active and doesn't trim context. Different commands may choose to operate on the navigation context in different ways, although a couple standard ways cover most of the cases.

As I'll explain in detail later, the web UI doesn't actually require you to make a selection before invoking a command. You're free to select first and then click a command, or click the command first and then make a selection. (If an appropriate selection doesn't exist when a command is invoked, you're prompted to make it at that point.) The selection mechanism and command verbs designed to give you lots of latitude about the order you do things when carrying out commands. Besides fitting better to your personal habits, this makes it harder to do something "wrong."

The web UI leverages its navigation network for uses besides the heuristics for cleaning up command navigation clutter. For example, the UI implements a history map view that is a generalization of today's linear history. It makes it easy for you to revisit a prior place, with important cues about the context in terms of other places you've were visiting at the time. The Next command works with the branching history too. You can explore a chain of links, back up, explore a different chain, back up, and Next your way down the original chain to get back to where you were. This is much easier than having to manually re-follow the original chain of links.

I think this is an important extension of the browsing metaphor. Unlike today's browser UI with its linear navigation chain, the web UI doesn't forget all the twists and turns of where you've been just because you back up and proceed in a different direction. It remembers not only where you've been, but also the path(s) you took to get there, and you can use the history map and/or the Next/Previous commands to get there again.

Network structuring and persistence, then, are key concepts of the navigational model. On a wireless-equipped device like the PV this is quite practical despite the PV's limited local storage capacity.

Given a network-style navigational context, you probably wonder how the Next function works. It works using heuristics to pick which path forward you probably intend. The most basic rule is to pick the forward path along which you backed to the current node. Other rules provide additional intelligence to account for your known navigational patterns, such as whether you got to the current node by navigating back by individual pages or by groupings of pages (such as by site), or by linking from a parent to a child. The Next function has options to present you a list of forward choices, textually and as a map in which you can zoom into desired areas; this way you can control which branch to take, if the default isn't correct.

Part of what makes the navigation and context trimming heuristics possible is built-in knowledge of logical levels of information grouping. For example, the UI understands about collections of pages making up sections and chapters, of collections of sections and chapters making up document, of collections of documents making up web sites, and so on. Likewise for the layers of command finding and invocation. Such knowledge of semantic clustering helps guide the decisions about popping contexts and for presenting the user reasonable choices about points to jump to along the Next/Previous chain.

Web UIs raise another general problem. Suppose you have fill-in fields like the TO: field of a message. Here you want to open the address book and make possibly several choices. How does that work within the document/navigation metaphor? Certainly forms can have special-purpose chooser controls for this job, and should where this provides a good shortcut for the most common choices. But the chooser UI should also make it possible to leverage the full power of the navigation, browsing, viewing, and search UI that is available in list-oriented documents like file folders and the address book. Making a choice from such a list should be a matter of just opening that list as you would in any other context and making the choice.

To take the address book example, the obvious answer is to have a link to the address book that's associated with the input field. But how do you get the user's choices back to the input field without this being a funny kind of modal mechanism or one that limits how the user can view the address book?

The answer is to make the link from an input field to its choice document be a command with navigation behavior, as opposed to being an ordinary link. This command navigates you to the document you need to choose from and captures anything you select. To make your selections, you use the normal selection idioms I'll describe later (or you can imagine the current desktop idioms).

Continuing with the address book example, when you're done selecting the addresses you want, what then? To finish, you use normal navigation controls to go back to the send form where the TO: field is. Alternatively, you can close the address book or use an OK command, either of which returns you automatically. There's nothing to save, because all choice state is captured as you go (if you change your mind, you can always cancel your current selection, or clear the TO: field when you return).

If you later want to change your choices, you click the TO: field link again and you're taken back to the address book, with all current choices still highlighted. This is because the command that takes you to the address book picks up the TO: state and paints the required selection regions.

So, you're using a standard document, in this case the address book, and the normal multiple selection idiom to make and change your fill-in choices. You have the full power of the normal UI for navigation, viewing, and so on when accessing the address book and can even navigate to other documents containing addresses where you can make other choices. Because the selection state is associated with the path you used to reach the address book, you'll only see the TO: items highlighted if you link to the address book (or successor nodes) via the TO: field in question.

This is an important principle of the web UI: behavior and presentation of a document can be affected by the path you used to reach it. It's a way to achieve stateful behavior without requiring special modes or UI mechanisms like dialogs. The implementation of chooser fields is one of the more important uses of this concept.

A final architectural topic central to the web UI is the model for creating and saving information. I'll discuss this in detail later, so for now just the essential points.

Any time you create something you, you're creating a new HTML document and linking it into a context. For an object inserted into an existing document, you're actually linking it into the document that is to contain it, with link properties implicitly set to make the object visualize in place (OLE-style embedding); physically, the object is stored as a child within the parent document's container.

For new, standalone objects, like a new Word document, the object is instead added to the current navigation context, as if you had done a Next to it from wherever you were when you issued the New command. Physically, the object is stored the user's sea of "free space", not part of any folder the user is aware of, unless and until the user chooses to file it somewhere (in reality, it is in a hidden, system folder).

You never need to put documents into the filing hierarchy or save them. This is because the web UI remembers all navigational history, and so you can always find the documents you create by viewing or searching the history map. You can of course file a document into a folder as an optional step, using a Save As command, and you can use Save / Save As to update or create versions of a document in the filing hierarchy as you please.

Using the selection, command, persistence, navigation context, and clustering mechanics I've described so far, it's possible to build the entire UI for a device like the PV or even the desktop, as will become clearer in the rest of this memo. Starting with just knowledge of how to page through a document and to follow links, a user can learn how to do any other UI operation. And this entire UI works without drop-down menus, toolbars, windows, or other cluttering widgets.

However, even if all commands were only one hop away, the web UI wouldn't be ideal, because you do want the most common commands to be zero hops away, and because you do want some context sensitivity to command presentation, like that provided by context menus. The web UI therefore accommodates things like toolbars, context menus, and other UI shortcuts that the user can customize—and does so by implementing them as HTML documents, like everything else. You can think of the UI shortcuts as being layered on top of the base web UI, yet they are constructed out of the same components that comprise the web UI: documents and links.

The focus of the next several subsections will be more on these shortcuts as they appear to the user, since they're what most users will use in their day to day activities with a PV. A user would learn the shortcuts when looking up commands in the operating manual, and so with time would branch off in to the web UI less and less often.

6.1.2 PV UI Specific Elements

The PV is made up of several elements that are at the center of the user's UI experience. These will all be explained in detail throughout the rest of this memo, but here is an overview:

- The case edges. The edges of the PV case are touch sensitive. The touch zones at the four corners are used to initiate navigation operations like turning pages. Labels may be applied to these areas to cue completely naïve users how to turn pages the first time they open a PV.
- The display surface. This is touch sensitive for both fingertip and pen operations.
- The display margins. Documents that present on the PV always show as print-like pages, including margins. The margin space is available for jotting notes and displaying UI-generated features like bookmark tabs and UI shortcuts. The default margin use is:
  - Top margin: bookmark tabs. Bookmarks or other objects placed here are associated with the containing document.
  - Bottom margin: document shortcuts. Document shortcuts or other objects placed here are associated with the display surface.
  - Left margin of leftmost display: clippings tabs. Clippings or other objects placed here are associated with the containing page.
  - Right margin of rightmost display: command shortcuts. Commands and other objects placed here are associated with the display surface.
  - Inner margin to right of spine (dual display PV): clipping tabs for that display page.
  - Inner margin to left of spine: not used.

  Because you can reassign which functions are associated with which margin, from now on I'll mainly refer to the margins via the names bookmark, document, clip, and command margin respectively.
- The display content. This is the area inside the page margins where content is displayed. Content may include links that the user can navigate by touching them, as well as user-created highlighting, jottings, and embedded notes and recordings.
- The note form. This is a document template that is used to implement several important PV features, including bookmarks, notes, and clippings. The note form is just a blank document that has predefined fields like an entry field for a title, and option boxes to control the note's presentation and behavior (e.g.,

its color, type, source back link, and anchor spec). The note form also has a Send button.

- The built-in documents. The PV has a few built-in documents, as listed below. Note that there is nothing "special" about being a built-in document; these are just documents that a new PV happens to contain by default. The list can be customized as desired, depending on the model of PV and it's intended application. The standard built-ins are:
  - Sign-in Page. This document provides the form for signing in when the PV is set for secure mode.
  - Start Sheet. This is the central starting point for the PV. From here you can link to all the content accessible to the user (stuff on the PV itself as well as in the user's desktop workspace and the Internet). In the ideal incarnation, the start sheet is the personal newsletter as described in my Beyond Browsing and Hyperactive Desktop memos.
  - My Documents. This is a folder listing all the documents you've read, authored, subscribed to, or purchased. I.e., it is the catalog of your personal library. It supports various views to let you get an overview of what you have and to find stuff you're looking for. This folder also has links to other PV content like the address book and operating manual.
  - Operating Manual. This is both the PV's help document and user interface. The UI is embedded in the operating manual as links that perform UI functions (web UI model). The operating manual has three associated documents that are of particular note: the settings sheet (preferences and config), the quick sheet (frequently used UI commands), and the status sheet (PV operating and communication status).
  - Map book. A book of maps, including the current site map, history map, topic (browsing) map, net neighborhood, physical vicinity (nearby machines), and local machine.
  - Annotations Folder. This is a folder of all the notes, clippings, bookmarks, jottings, and highlights you've entered into your various documents. These items are all stored as external annotations in the Annotations Folder. You don't usually view this folder, but instead view one of the following persistent views derived from the Annotations Folder.
    - Clippings Folder. A persistent view of the Annotations Folder, showing only clippings. The default view categorizes by document, with more recently read documents ahead of less recently read.
    - Bookmarks Folder. Same as the Clippings Folder, but showing only bookmarks.
    - The Notes Folder. Same as the Clippings Folder, but showing only notes.
  - Notebook. This is a blank document where you can write notes that are not associated with a particular document. There's nothing special about the notebook—it's just provided by default since most users would want to have a notebook to write in anyway.
  - The mailbox, calendar, and address book. These will sync with your desktop- or network-based counterparts.

PV's can exist in single- as well as dual-display models. Some UI actions on a dual-display model will cause only the rightmost display image to be replaced. When this happens, the original image may or may not be shifted to the left display when this happens. The rule is that when the UI action was invoked from a link or menu action initiated on the right display, the image is shifted to the left display and the new page opens on the right. If the action was initiated on the left display, the new page simply opens on the right and no shift occurs. The result is that the new page always opens on the right and the page from which it was initiated always appears on the left.

For a single display PV, the current display image is simply replaced by the new page. You can always use the Previous and Next functions to flip between the original and new pages.

PV's can be held horizontally or vertically. Further, dual-display PV's can treat the two displays as separate pages, or as halves of the same (large) page. The PV always adapts its assignment of case touch zones and display areas to present a consistent geometry to the user regardless of its orientation and display mode. That is, the PV re-maps its definition of things like "upper left corner" and "right margin" to be consistent with its current orientation.

All UI actions on the PV occur via touch operations using the fingertip or pen. You can use a fingertip or the pen interchangeably. The pen has three tips: the erasing tip at one end, the highlighting tip at the other, and, by twisting the pen barrel to extend a small shaft, the writing tip. The fingertip is recognized as a highlighting tip, with a gesture allowing it to erase.

The basic touch idioms are:

- Tap (touch and release after a brief time). The action taken depends on where you tapped (see table below). It does not depend on what tip you are using. The tap must last a certain minimum (configurable) time, so that an accidental or glancing touch will be ignored.
- Hold (touch and hold contact). The action taken depends on what you held; usually it opens an object or its context menu (see table below). The action does not depend on what tip you're using. The hold time is a preference parameter.
- Swipe. A swipe is any motion of a tip in contact with the display surface. Swiping will write or erase ink or highlighting, depending on which tip you're using. Swipes are also used to make and extend current selections.
- Tap-swipe. A tap immediately followed by a swipe starting at the same spot. This causes the pen or fingertip to start a current selection region. Either the writing or highlighting tip can be used. A selection can also be drawn by doing a swipe starting at the current edit cursor location (i.e., the tap is optional when starting on top of the edit cursor).

- Erase (a jiggling swipe). When done with the fingertip, this is treated as a swipe with the erasing tip. (The motion is the same as erasing with a real pencil eraser). With the pen you can just swipe with the erasing tip.

You can automatically extend the range of a highlight, current selection, or erasure by holding the pen or fingertip after swiping a part of the range; that is, swipe part of the range, then hold for auto-complete to take over. The range auto-extends to end of word, sentence or line, paragraph, page, section or chapter, and document, in that order, the longer you hold. The UI provides feedback on the extent of the selection. In tables, the selection auto-extends by cells to end of row or column, and then by rows or columns, depending on whether the swipe was across or down. Arbitrary rectangular blocks can be highlighted, selected, or erased by swiping a box shape around the desired area. Selecting the page number of a page selects the whole page. Selection mechanics and options are fully explained in Section 6.5.1.1.

Note that everything you can do with a pen you can do with your fingertip, except write ink. There are also keyboard/mouse idioms for all the pen idioms and pen tips:

- Tap is left-click
- Hold is hold left button
- Swipe is hold left or right button and move mouse. Swiping with left button down selects, with right down highlights. There is no need for tap-swipe.
- Erase is hold both buttons and move mouse
- Jotting is done via the keyboard (i.e., you have text instead of ink)

The mouse supports a few other idioms as follows. Note that these and all the other idioms can be customized via the Settings Sheet.

- Right click brings up a context menu, the same one that a hold operation at this location would.
- Mouse wheel does page forward and back operations.
- Clicking the mouse wheel / third button links to the Quick Sheet.

The following table is a "quick ref" for some of the most important UI shortcuts of the PV. It's intended only to give a kind of gestalt feel for what's involved in using the PV. The subsequent sections will explain all these functions in detail, their various options, and other PV operations not covered in the table (such as the editing and collaboration functions).

Area Touched:[1]	Tap	Hold	Erase	Swipe[2]
Upper left corner of PV	Previous page in doc	Riffle pages back	N/A	
Upper right corner of PV	Next page in doc	Riffle pages ahead	N/A	
Lower left corner of PV	Browser-like back	Present IntelliView Navigation choices	N/A	
Lower right corner of PV	Browser-like forward	Present IntelliView navigation choices	N/A	
Any screen area, inside selection	Cancel selection	Open context menu	Unselect the area erased	
Ink or highlight or both, in any area	Play recording, if any, if AutoSeek is on[3]; otherwise do per what exists underneath	Do as stated elsewhere in the table for what's at this spot	Erase as present & editable, in order: highlights, ink, link, underlying content	
Empty space in top display margin	Insert bookmark	Open context menu	N/A	
Empty space in left display margin	Insert clip	Open context menu	N/A	
Empty space in right margin	Insert command shortcut	Open context menu	N/A	
Empty space in bottom margin	Insert document shortcut	Open context menu	N/A	
Bookmark tab	Go to bookmarked page	Open bookmark[3]	Delete bookmark	
Note icon	Play contained recording, if any, if AutoSeek is on[3]	Open note[3]	Delete note	
Clipping icon	Play contained recording, if any, if AutoSeek is on[3]	Open clip[3]	Delete clip	
In display content on hyperlink	Follow link	Link-specific action (normally: open link preview info)	Erase link if editable	
In display content between lines or in white space	Insert note	Open context menu	N/A	
In display content within a line	Insert edit cursor	Open context menu	Erase characters if content is editable	
Page number	Go to document TOC	Open context menu	Erase as present and editable, over whole page, in order: highlights, ink, links, content	
UI icons or command links	Perform UI command	Command specific (normally: open context menu)	Erase from display if permitted	

Note 1: The table lists UI idioms in priority order, meaning that in cases of overlap, the action specified by the first applicable row is the one that occurs. For example, tapping within a line in a selected area is both a case of tapping inside a selection and tapping within a line. Based on the ordering of the table, this is recognized as tapping in a selected area, so that selection is cancelled (selections don't care where inside them you tap).

Note 2: For brevity the above table doesn't include a column for swipe actions because these are mostly dependent on the tip being used, not the thing being swiped over. That is, swiping will simply draw highlights, write ink, erase, or extend a current selection, based on the conventions described earlier in this section. You can highlight, jot, erase, or select anywhere on the display, including any of the margins. Note that erasing follows a priority order when things overlap. For example, if you've highlighted a jot, then erasing takes out the highlight first. Erase again to remove the ink. Erase again to erase any (editable) underlying link. Erase again to remove any editable content. Erasing a box around something and holding will erase everything in the box in a single action.

Note 3: The operation of the IntelliTape recorder can affect the behavior of notes, clippings, and bookmarks as follows:

- When the recorder's AutoSeek feature is on, tapping a note or clipping icon will automatically initiate playback its contained recording, if any. Notes, clippings, and bookmarks containing recordings will also automatically play when opened if AutoSeek is on.
- When the recorder's AutoRecord function is on and you tap to insert a note, the note is inserted as a cassette icon and recording is automatically engaged the next time you speak.
- The icon for objects containing recordings are embellished with a cassette figure.

AutoSeek also affects tap operations inked or highlighted areas: playback occurs of any comments recorded while viewing that page, or when that area was inked or highlighted.

See Section 6.4.6 for a detailed explanation of the operation and behavior of the recorder.

Note 4: If on page 1 of the document TOC, tapping the page number takes you to the Start Page.

6.2 Starting the PV

You turn on the PV by opening it up.

At this point, the PV has two decisions to make: should it require you to sign on, and then what content should it open on for you?

6.2.1 Signing On

PV's support a sign-on function that lets you identify yourself. This is important for your security, because the PV provides access to your workspace (either through a wireless modem, or via the IR link to your desktop machine as described in Section 6.8).

Sign-on also makes online purchasing convenient, because the PV can hold name, address, and charge card info that is automatically transmitted to vendors—but only if you've signed on. Just as important, sign-on makes the PV useless to a thief. Without a successful sign-on, the PV won't start, and can even be set up via preferences to sound an alarm and send an email, a page, or other kind of communication alert when thievery is detected. (Sorry, the electrified case option needs to wait for new battery technology.)

Sign-on occurs at two levels. The first level, local sign-on, is applicable to all models of PV. It "unlocks" what's contained in the PV's memory, giving access to the content that's cached there.

The second level of sign-on is network logon. Network logon only happens for PV's equipped with wireless modems or that are in range of a machine that acts as an IR router (such as a desktop machine; see Section 6.8). Network logon uses normal authorization protocols to log you on to your ISP or corporate intranet. Preference settings control in what situations to do network logon, and to what logon authority the logon should be directed.

The existence of local and network logon as separate levels must of course be transparent. You just sign on and the right thing happens at the right time. The PV contains a status sheet (Section 6.6) that gives details about your logon (and other) status in cases you really need to know that.

So how does sign-on work in a device that's not that input-oriented?

For PV's without pen input, the sign-on page brings up a keypad for entering your password. For PV's with pen input, you could print your password or ideally just sign your name. The former requires just print recognition, while the latter would need signature recognition, admittedly a very advanced feature. However, while it's beyond the scope of this memo to discuss, I believe a way can be devised to enlist the participation of the authorization server to perform the hard part of signature verification as part of the logon protocol, and to do so in a way that doesn't compromise the security of the signature (such as by sending it in a way that others could copy it).

Somewhat farther in the future, PV's may come equipped with fingerprint or face recognition. Recently, integrated fingerprint recognizers have become available for a couple hundred dollars, so a very cheap version may be only a few years off. Face recognizers will become practical when PV's can be equipped with integrated two-way video.

The PV needn't provide an explicit sign-off procedure, because you can accomplish that just by closing the PV. Depending on preference settings, the PV can log off immediately on being closed, or after a set time delay. Some people may like the delay option, because it gives them the flexibility to close the PV for a while and open it without needing to do another sign on.

6.2.2 IntelliView

When you open a PV, and after a possible sign-on step, the next thing to happen is for content to appear. What content?

The simple default is to show you whatever you were reading when you last closed the PV (or the operating manual's Getting Started page if this is the first time you've used the PV.) For many cases, this is exactly what you want. But not always. Sometimes you're opening the PV to look at something else.

To be smarter about what content to open for you, the PV has a feature called IntelliView. IntelliView uses heuristics and/or explicitly customizable rules to guide the choice of what to bring up when you open the device. The scenario in Section 5 showed an example of this, when the PV brought up the morning newspaper, which isn't what I'd been reading the night before.

The IntelliView heuristics are mainly based on the your usage pattern as referenced to the time of day, as well on detection of other machines in the vicinity. It's fairly easy to deduce that the newspaper should be shown if the PV is opened early in the morning, because I almost always open the paper at that time.

Likewise, it's easy to deduce that an hour or so later the PV should show whatever's on my office desktop screen, because by then I'm usually in my office and using the PV to augment reading what's on the desktop screen. The PV can notice that, indeed, that time has arrived, and my desktop PC is in the local vicinity.

In the presence of a machine showing a presentation, such as in a meeting room, I'm usually interested in watching the presentation locally. The PV can know about this based both on detecting a presentation being shown on another local machine, and based on my calendar information that says I'm in a meeting.

Some of the more straightforward heuristics, like the ones I just described, can be built in to the PV and made customizable via the settings page (Section 6.6). The PV can then fine-tune the heuristics from there, based on experience. We can also provide a way for people to define custom rules that drive the IntelliView heuristics.

And what if the heuristics make the wrong choice? For example, one morning I'm sick and instead of reading the newspaper I want to stay in bed and continue reading the novel I started last night? Simple, when I open the PV and the morning paper shows up I just use the "previous" operation (Section 6.3.4) and I'm back where I left off in my novel. This works because IntelliView is just an automatic navigation operation. Alternatively, if what I need to see is something else entirely, I can explicitly navigate to it (Section 6.3.4).

IntelliView logic comes into play when using several other commands where users are given navigation choices—for example, the Next and Previous commands (Section 6.3.4), Undo/Redo (Section 6.5.2.3), and Save As (6.5.2.4). IntelliView heuristics also control the dynamic appearance of command shortcuts and toolbars in the margins (for example, see Section 6.5.2.5).

6.3 Reading

Because the primary activity on a PV is reading stuff, I'll start with the essential UI components for reading. This entails the way that pages are presented and the ways that you turn them, skim, navigate, use bookmarks, and take advantage of the PV's different display orientations.

6.3.1 Page and Document Presentation

A goal of the PV is to make the reading experience as much like reading printed matter as possible. To that end, the PV eschews the typical computer-like model of presenting information in windows, as a scrollable stream, with visible UI appurtenances. Instead, the PV presents information in a print-like, page-oriented way—literally "what you see is what you print". Depending on the PV model and orientation, you see one or two full, print quality pages, with appropriate page elements like margins, headers, footers, and page numbers. It is an image that can be printed in true WYSIWYG manner.

Furthering this print illusion is the use of subtle detail, like a background texture suggestive of paper. Page turning may be animated to simulate the appearance of a real book or document page being turned, and this may be accompanied by a matching sound effect. Preference items let you customize or disable these behaviors.

Much of the material PV's access will already exist in page-formatted form, so dynamic page layout won't typically be necessary. (Section 7 discusses the file formats question in some detail.) Of course, to handle existing stream-oriented files, the PV will be able to do the necessary page formatting, or at your option, display stream information in scrollable form.

Most of the documents your read on a PV will be longer works, like books, magazines, product specifications and plans, etc. Such documents have structure like chapter and section divisions, a table of contents, indices, appendices, glossaries, or some combination of these elements. If the content file is coded appropriately, these manifest in ways that provide appropriate interaction facilities. For example, items in the TOC and index are rendered as links to those parts of the document. Inside the document, chapter and section references and terms in the glossary are also rendered as links. Page numbers on each page link back to the TOC. The PV also recognizes elements like the TOC and provides explicit UI commands in the web UI like a "go to TOC" function.

When reading, there will be no UI controls of any kind visible. You may, however, elect to keep certain controls visible as shortcuts so that they're quickly selectable via a touch operation. When visible, shortcuts appear in the command shortcuts margin (default: right margin). Examples are shortcuts that take you directly to standard parts of a document, like the index, or that navigate by document divisions (next/prev section), or that request editing control of the shared workspace in a meeting.

All commands that can be manifested as shortcuts are of course available via the web UI (that is, as hotspots on a page). Any web UI command can be made a shortcut. Doing so is just a matter of copying the command off the page to the shortcut area, as explained in Section 6.7 (customization of command parameters is also supported).

6.3.2 Page Turning and Riffling

Page turning is the primary activity of any printed reading material. The PV hardware makes page-turning as "paper-like" as possible by using the corner zones of the PV as the page turning hotspots. For example, to advance a page, tap the upper right corner. To go back a page, tap the upper left corner. This is much like the way you'd turn pages in a physical book (you grab a corner of a page). The PV is like a book with motorized pages.

You can also page through the material a section or chapter at a time. These functions and other related ones are available on the Quick Sheet and in context menus as explained in Section 6.3.4.

The PV's corner assignments are configurable via the Settings Sheet (Section 6.6). For example, some users may prefer that the lower corners control page turning rather than the top ones.

The PV adds a virtual page to the front and back of every document. You get to these pages if you page past the first or last real pages of the document. Their purpose is to assist you in navigating at the document boundaries. Both pages provide a set of links synthesized by the IntelliView logic that represent good guesses about where you'd likely want to go next. The guesses are based on the record of document-visiting associations you've made in the past; i.e., after visiting the stock quotes web page, Fred usually goes to the financial news page. Or, after reading Time Magazine, Jane usually reads The Economist. The heuristics may also be modified by time of day and other factors that have strong associations to your navigation patterns.

Besides the IntelliView guesses, the virtual pages include links to standard places like the Favorites Sheet, History Map, Start Sheet, My Documents, and Search (see Section 6.6.). There's also a command button that lets you add a add blank page here in the document for you to take notes on (see Section 6.4.2).

Skimming is another important reading activity. It is just a fast form of continuous page turning. One way to accomplish this is to hold a corner rather than just tap it. While holding it, pages (or whatever) will advance at a preference-settable rate. So, holding the upper right corner riffles ahead by pages, holding the upper left riffles back. To speed the riffling rate up, slide the finger away from the corner, maintaining touch contact, while the pages advance.

When in two-page mode, the PV supports two visual styles for page turning, selectable via preferences. One is the true book-like style of showing you two new pages; literally, you've "turned" to the next or previous pair of pages. The other style simply shifts you in the page stream one page to the left or right. For example, when advancing a page, the former right page becomes the left page, and a new right page appears.

6.3.3 Bookmarking

Being able to place and return to bookmarks is an essential reading activity and the PV supports this.

One thing it does for you is to automatically bookmark the last place you were reading each time you leave document. The next time you return to the document, the PV opens it to this bookmark. The user never actually sees this bookmark.

You can also place and open to bookmarks you've placed explicitly, and which you do see. To place a bookmark, you tap anywhere in the bookmark margin. A bookmark tab will appear in the margin, showing the current page number.

Other, existing bookmarks will also be shown as tabs in the bookmark margin. Each displays the page number it is on, and an optional label. Bookmarks are basically hyperlinks. To go to a desired bookmark, tap it. Note that unlike turning a page, navigating by a bookmark *is* treated as a linking operation; so you can use the Previous operation to return to the place you bookmarked from (see Section 6.3.4.2).

Bookmarks have properties and a content area for jotting notes that you can manipulate. To open a bookmark, you hold its icon. The bookmark's content page will appear on the right display of a two page PV. If the page containing the bookmark was on the right display, that page shifts to the left display. On a single display model, the current page is simply replaced by the bookmark page.

The bookmark's content page gives you a space where you can jot notes on the bookmark, plus there are controls for deleting the bookmark and for accessing the bookmark's properties, like color and texture. You can also delete bookmarks and get their properties using the normal selection/context menu mechanism (Section 6.5.1.2), and you can erase them directly by erasing over the bookmark with the pen or fingertip (Section 6.4.1).

The bookmark content page is actually a version of the general note-taking page. All the features and behaviors of notes as described in Section 6.4.3 apply to bookmarks. For example, this includes the ability to capture multiple pages of written notes as well as voice notes on the bookmark, and to insert the bookmark so that it appears not only in the margin, but also at particular spot in the text. See Section 6.4.3 for more details on this.

6.3.4 Navigating

Page turning and riffling are the only ways to navigate a physical document, but not so for electronic documents. The PV can offer a much richer reading experience by giving users additional navigational and browsing controls. Navigation facilities include hyperlinking, Next/Previous, Favorites and History, the Quick Sheet, and Search.

6.3.4.1 Hyperlinking

One way to navigate is by hyperlinking. The PV renders textual hyperlinks using a visual emphasis just like a browser does. Tapping a link will navigate you to that place. You can use the Previous and Next operations to move along the link chain. The PV Start Page always exists as the most previous in the chain.

You can also hold a link. Doing this will perform a link-specific action. The default behavior for hyperlinks is to present preview information about where the link will take you; i.e., document name and document position information and/or a thumbnail. Continuing to hold the link will cause the preview information to expand into a navigational map of the link context emanating from the link you are holding.

When you release a held link you will not navigate; you must tap it to do that. Note that if holding the link opened a navigation map, the map will persist for a short time after you let go. This gives you a chance to tap a spot on the map to go there, or to hold a spot in order to preview and then to expand the map context around that point.

Many elements of a document will act as links, including TOC and index entries, page numbers, section and chapter references, and glossary terms in the text. Image-mapped links are supported. Documents can of course also include authored hyperlinks, as well as user-created hyperlinks (see Sections 6.4.7 and 6.5.1).

Note that because you normally use the PV without a mouse, you don't get the advantage of a hand cursor to show you where links are. You must instead rely on the visual emphasis that link hotspots provide. This works fine for text, but it rather depends on the author for making it clear whether a given image or part of an image acts as a link. Of course, the user can always hold a spot to determine if it is a link; if preview information comes up, a link is present.

6.3.4.2 Next and Previous

Another way to navigate is via browser-like Next and Previous commands, performed (in the default assignment) by tapping the PV's lower right and left corner's respectively. Specifically, these functions navigate you along the chain of visitations caused by link operations.

Note that page-turning operations are not part of the next/previous chain. For example, if you open a book, read the first ten pages and then do "previous", you will return to wherever you were before you opened the book; you will not go back to the previous page of the current document (for that, you use the previous page operation). The distinct page-level and navigation-level functions exist because of the PV's scroll-less UI. The page forward/back functions take the place of the traditional scrollbar for scanning within a single document.

The Next and Previous commands implement IntelliView logic much as I described above for the virtual first and last pages in Section 6.3.1. This feature is invoked in two cases: 1) automatically, when you Next/Previous past the end of the navigation chain, and 2) manually, when you hold (rather than tap) the Next or Previous button.

In either case, a list of IntelliView guesses appears for you to choose from. These may be similar but probably not identical to the choices that would appear on the virtual first/last pages of the current document, because the navigational context of the choice is different (you're not at the start or end of the document).

For Next, if you're not yet at the end of the navigation chain, the first item in the IntelliView list is the next item in the chain, followed by the IntelliView guesses. (If the chain branches at this point, the first will be the next item along the newer branch.) The guess items may be but need not be other items further along the navigation chain. For example, if the last navigation function you performed was "next section" within the document, then a good guess for the list is again "next section", especially if you did the "next section" operation within the last few seconds. Or, if experience teaches that from this point you almost always visit a specific other document, that document would also show up as a guess.

In the majority of cases, three or four IntelliView guesses is enough to ensure that one of them is where you want to go next. The last item on the Next list is always Favorites. This choice navigates you to the Favorites Sheet (see next section).

For Previous, the list of choices includes the immediately previous item in the navigation chain, with a list of three or for guesses for other (more previous) places in the chain that you might want to go do directly. The guesses usually help you go immediately back to some previous context (e.g., site) in the chain, without having to Previous over a bunch of intermediate pages that were part of navigating an intervening context. For example, if you previously linked from the Microsoft site to Yahoo and wormed your way around Yahoo for a bunch of pages, then one of the previous list guesses would return you directly to Microsoft, thus avoiding the need to back through all the Yahoo pages.

The last guess on the Previous list is always History. This choice navigates you to the History Map (see next section).

When viewing the IntelliView guesses, pressing the next or previous button a second time will navigate to the first item in the list.

6.3.4.3 Favorites and History

The PV maintains a traditional favorites list as a document you can link to and view in a variety of ways. One way to link to it is via the Next button as described in the last section. Another is via the Quick Sheet, as described in the next section. Or you could put a Favorites command shortcut in either shortcut margin as described in Section 6.7. Tapping this shortcut takes you to Favorites. Holding the shortcut brings up list of options, which include a verb for adding the current document/site to the Favorites list.

Note that when you link to the Favorites Sheet via the Favorites command, the sheet will take itself out of the navigation chain as soon as you select from it. Being a command, Favorites cleans up the navigation context as described in Section 6.1.1. On the other hand, if you were to access the Favorites Sheet by following a non-command link to it—you could create one if you wanted—the sheet would *not* remove itself from the navigation context. (Section 6.5.2 explains how to create links.)

For history, the PV maintains a History Map that you can link to via the Previous button, the Quick Sheet, or a History command shortcut, much as for Favorites. More than just a listing of places you've visited, this map includes information about the order in which you've visited things. That is, it preserves navigational history in the form of a network that you can view graphically. A variety of views let you look at information textually, too.

The graphical view of the history map uses knowledge about semantic clustering to present the map at a logical level of detail—for example, as nodes representing web sites you crawled around in, or as documents within sites or folders that you browsed. By tapping a node you can return to that location, or by holding a node you can expand the map's detail level at that point for a more detailed overview of the history around that point. This works just like viewing and expanding link preview maps as described in Section 6.3.4.1. For example, holding a web site node would expand it into the history of web pages you've visited within that site.

The textual views on the favorites and history information include the traditional sorts, filters, and categorizations on various properties like type, creation timestamp and name. For example you can view either list in order of most recently visited, or categorized by month/year and filtered for site names only, or categorized by type and sorted by name. Filters and categorizations on type are especially useful for letting you separately view your favorite web sites, documents, newsgroups, and chat rooms. An additional property keeps track of your reference frequency, so you can sort on your most frequently visited favorites or history too.

The Favorites and History documents share a default organization that is like other list-oriented documents, including for example file folders and mail. Specifically, there is a TOC whose section headings correspond to various common, canned views on the list, and an index that provides an alphabetized listing of the items in the list—in this case, a two-level index of document/web page within folder/site. The documents also include predefined bookmarks for each section of the document. To change views, you can visit the TOC and click on a section heading, or click on one of the bookmarks. You can also access views via the Views command as described in Section 6.3.8; this lets you choose among the canned views as well as create and customize them.

The organization for list-oriented documents is discussed further in Section 6.3.8.2.

6.3.4.4 Quick Sheet

The PV provides a rich set of navigation facilities via a document called the Quick Sheet. The Quick Sheet contains shortcuts for the most commonly used UI operations, including navigation operations. Think of it as a quick ref card for operating the PV. The full range of what it contains and how you can customize it are discussed in Sections 6.6 and 6.7. Here I'll just summarize its navigation commands.

You can access the Quick Sheet in several ways:

- Tap a shortcut to it that exists in either shortcut margin. If one hasn't been put there, you can create one simply by tapping any empty spot in the margin. A command shortcut will appear.
- Hold the page number at the bottom of either display page to open a context menu. One of the menu choices is Quick Sheet.
- Page to the front or back of the current document (the virtual first or last page as described in Section 6.3.1). A Go To Quick Sheet item is one of the choices on the IntelliView list.

Visually, the Quick Sheet will appear in the right display of a two-display PV. If you invoked it by a UI action on the right display, that page shifts to the left display so it will remain visible.

The Quick Sheet may include all or a subset of these navigation functions:

- Browser-like Next, Previous, History, and Favorites commands (see Sections 6.3.4.2 and 6.3.4.3).
- Next/Previous Chapter, Section, and other pertinent divisions of the current document
- Page and line up and down, if the content is displayed in scroll mode (see Section 6.3.6)
- Go to Current Selection
- Go To Page Number (with touch keypad and pen fields for entry)
- Go To Table of Contents (of current document)
- Go to My Documents, Map Book Operating Manual, Start Sheet, Settings Sheet, Status Sheet (see Section 6.6)

- Search (takes you to search form—see below)

Of course, to avoid presenting too long a list of choices, the navigation commands may be grouped using drop-down lists or links so that only the most common choices are presented at the top level.

Note that the above navigation commands happen relative to the document you linked to the Quick Sheet from. In other words, it's as if an implicit "previous" is done first. The logic that takes care of this is part of the Quick Sheet command that links you to the Quick Sheet.

6.3.4.5 Search

A search operation is available, even though searching isn't convenient on a device lacking a keyboard. To accommodate it, the Search Sheet has a pen input field for inputting keywords, as well as a rendered keyboard that can be typed on with touch operations. On smaller PV's, the keyboard may occupy one of the two display screens, making search a two-page form in this case.

Search leverages fulltext content and property indexing of local stuff and stuff in your network workspace, plus it acts as a front end to web-based search engines. The Search Sheet has options settings for you to specify the desired scope of the search (current document, local machine, my workspace, my company, whole world, as well as user-specified URL/pathname).

Because the PV's web UI treats content and UI pages the same, search lets you find command and UI information the same as for finding anything. You needn't do anything special to trigger this, because the Operating Manual is indexed like everything else; just enter the search terms you're looking for (e.g., command name or activity). The user assistant notices when you're searching for UI-related content and may awaken to offer suggestions along with the search results.

Search results come back as a document, like everything on the PV. You can use a full complement of canned and customizable views to locally filter, re-sort, and categorize the search results. The search document is organized the same as I described for Favorites and History (Section 6.3.4.3). You can also annotate the search results if you like, and file them away, as you can for any document.

6.3.5 My Documents

One of the more useful destinations offered by the Quick Sheet is My Documents. This is a compendium of all the documents you've read, authored, bought, or subscribed to (actually, it is a folder of mainly links to these documents). You can think of it as the catalog of your personal library. It encompasses everything that the existing desktop folder of the same name does, plus more, because it's not just stuff physically contained on your PV—it's a record everything you've ever read or received (subject to garbage collection of state older than a cutoff).

For PV's with connectivity to your desktop or server workspace, My Documents will include all documents in those respective scopes, not just stuff stored or accessed from the PV.

My Documents supports a variety of custom views designed to help you get an overview of what's in your library, to see what you have and haven't read, to determine what stuff exists on your PV or in other storage locations, and to find things you're looking for. The views support the traditional set of folder viewing features, including filtering, sorting, and categorizing, based on properties like title, read/unread, subject keywords, and so on. In all cases fields like keyword entry fields are implemented as chooser lists so that text input isn't necessary.

My Documents shares the same kind of document organization as I described for Favorites and History (Section 6.3.4.3). This means you can easily access the canned views via the TOC and predefined bookmarks. The Views command (Section 6.3.8) lets you customize or define your own views.

6.3.6 Scrolling

Even though the PV is page oriented, there are times when a scroll-oriented presentation is the only reasonable display choice. The PV supports this by displaying traditional scroll bars when stream-oriented presentation is required. You use touch operations on the scroll bars to perform the scrolling actions. This looks and works exactly like the desktop equivalent.

6.3.7 Page Size and Orientation

PV's can be used to read a wide range of material. Because a goal is to equal the print experience, the PV needs to render things from small paperback books to normal office documents to newspapers, magazines, and oversize picture books—and to make these things look are read as much like the originals as possible, on any size PV. This means the PV needs to be smart about how it renders different kinds of content, depending on the size of the PV and in what orientation the user wants to hold it.

That being said, it will be unrealistic to get good results in some cases. For example, the largest-format high color electronic book will be readable on the smallest PV, but the experience may be marginal. This is one reason why users will often own more than one PV (e.g., a big color one and a small b&w one).

For a single-display PV, there are basically two presentation decisions that need to be made: should the content be displayed in the portrait or landscape orientation of the PV, and should the content be scaled to fit the PV display versus be repaginated or scrolled.

For a dual-display PV a few more choices are available. The PV can be used horizontally, with two pages displayed side-by-side in portrait mode, or a single, large page displayed in landscape mode (spread across the two displays). Or the PV can be used vertically—one display over the other—with two pages displayed in landscape mode, or one large (spread) page displayed in portrait mode. Imagine the various ways a publisher might adapt different kinds of art or photographic material to a fixed book format, sometimes using portrait or landscape orientation on different pages, sometimes (but rarely) spreading a single image across two facing pages. The PV permits authors to use these different formats, in different documents and within the same document.

Finally, a dual-display PV can be folded over (displays on opposite sides) for viewing a single, regular-sized page at a time, in portrait or landscape mode. An example of where this may be useful is an 8.5 x 11 dual-page PV being used to read a typical office document. This is because it's easier to a larger PV in the folded-over configuration, and office documents are usually read a single page at a time anyway.

The right page size and orientation to use is going to depend on the content being displayed. To let the PV make the right decision, the content needs to encode information like the desired page dimensions and orientation, and the range of reasonable scale sizes. The PV also needs to take into account factors like how large the gap is at the seam between the dual displays (in some models this may be minimal, in others much bigger). If the gap is too big, the large-size page format may not display well enough to be usable.

The PV will by default try to bring up information in the way that provides the highest fidelity to the content file's stated print format. So, if the file is an electronic novel optimized for 5x7 pages, and the PV displays are that size or close enough where scaling is OK, then the PV will open the book in the dual portrait-page mode. On the other hand, if the file is an oversize picture book using 11x14 pages, and this is a 4x5 PV, where 4x5 is outside the scaling range, then the PV may instead open the book in the large (spread) portrait or landscape mode.

If no combination of scaling and orientation will work to produce a legible result, the PV will resort to either repagination or a format that uses scroll bars. (Repagination is avoided because this compromises the publisher's design, of which pagination is a part.)

What if the user is holding the PV in a different configuration or orientation than the one the PV chooses to for displaying? Well, in most cases the user will just reorient the device as necessary and proceed with reading. For stubborn readers, the PV will have sensors to detect how the device is being held, and if it's not reoriented fairly quickly, it will re-render to adapt to the user's apparent preference (even though this may be less legible). The user can always reorient the PV if the result is poor.

6.3.8 Views

The PV offers a great variety of viewing options for material it presents. This is especially true for list-oriented material like collections of mail, documents, schedule items, search results, favorites, history, and so on. Many views are textual, some hierarchical, some graphical.

Views, as I've written many times in the past, are based on object properties and indexing. The property index I'm assuming here is as I've described for integrated storage: it is an index that can catalog the properties of heterogeneous collections of objects across an entire network scope, with efficient sorted and categorized queries, and permissions-based filtering on the server end of the query results. My Storage Unification memo explains how this can all work, and my Hyperactive Desktop memo discusses a lot of the more useful views over integrated storage, especially web-oriented views. I won't duplicate all that material here.

What's relevant here is to discuss how you choose viewing options using the PV's web-oriented UI, and how view design differs to fit in well to the windowless, scroll-less UI model of the PV.

6.3.8.1 Accessing Views

You access views the way you access all other commands. Views are implemented as commands that are hosted on the Quick Sheet and in the Operating Manual, in certain context menus, and that you can also place in the shortcut margins as well as in document content.

Let's start with the Quick Sheet. Besides the navigation commands I've already discussed, the Quick Sheet has a section for views. This is implemented as a "quick list" of the most commonly used canned views, plus a link that takes you to an advanced Views Sheet

The first entry on the quick list of views is the Current View. Choosing this opens a View Sheet that lets you change the settings for the current view.

The rest of the views quick list is dynamic according to what document you're currently looking at and which views you most commonly use in conjunction with it. The Views Sheet provides a full listing of all applicable views, plus commands that let you

- Add and delete views
- Customize existing views
- Perform one-off views using specified select, sort, categorize, and filter criteria

You can also get to views via context menus. Views commands are included on the context menus for page numbers and for selection regions that span an entity with viewing options—for example, an entire selected document, or any selected page of that document. What's carried on the context menus is the top few items of the views quick list, plus a More Views… command that takes you to the advanced Views Sheet.

Content that's very views-oriented, like mail and My Documents, have viewing commands authored directly into the content (see next section), so you can switch views without opening a context menu or views UI.. You can also add your own command shortcuts for views to the shortcut margin (see Section 6.7).

To activate a view, you tap its command hotspot. Holding a view command opens that view's View Sheet, which lets customize the settings, either one-off or permanently (if permitted). The View Sheet also carries explanatory text about the view.

6.3.8.2 View Appearance

Today's desktop views are very pane- and scroll-oriented. It's common to have one or more panes for making choices according to information content and/or hierarchy, plus a window where content is shown. All panes are scrollable.

While the PV can support multipane scrollable views, the preference is to use views that depend instead on links and in-place expandable/collapsible hierarchies (instead of panes) with page turning instead of scrolling. Why? Because I believe this results in an easier to understand and less cluttered presentation for most users; it also better leverages the document-like UI mechanisms that all the rest of the PV UI is based on. The best way to explain this is by example.

Consider the typical file explorer view, which has its counterparts on the PV in views for folders like My Documents. Instead of a two-pane view with hierarchy and content, My Documents provides a more document-like view that does the same thing.

In this view, My Documents appears literally as a document with a table of contents, content pages, and an index. The TOC gives you listing of various common views on My Documents, presented as "sections"—the content—of My Documents, while the index gives you the flat list of the children of My Documents. A set of predefined bookmarks also mark each of the document's sections (i.e., canned views) so you can visit them without flipping to the TOC.

One of the sections, for example, is the alphabetical listing of the children of My Documents, with item details like read/unread status, title, author, and modified date, all arranged tabularly. Another section gives each child item along with inline preview information, much like the Outlook preview view. Another section is the icon grid view, in which children show up as icons you can move around.

There's also a section that presents the hierarchy My Documents is a part of, in the form of an expandable/collapsible outline; this is useful if you want to navigate elsewhere in the hierarchy. Other canned views available as My Document sections provide additional tabular, categorized, and graphical views of the folder. These are all described in Section 6.6.

Because My Documents is presented as a document, you can do all the normal highlighting, jotting, annotating, and editing functions that you can with any document. For example, this means you can add notes to the page, prettify them with additional text and images, and directly manipulate document properties by editing them in place in the views.

The Favorites, History, and Search results documents discussed earlier are other examples that use the above organization.

If you want a view not provided by a folder's TOC, you can use the Views Sheet described in the last section to do a one-off. Or, you can insert a new view section by copying an existing one and customize its settings as you please. Note that the list of views carried by a folder also appear in the list of canned viewed offered on the Quick Sheet (see last section), where the section heading is the name of the canned view.

As you'd expect, you can go to a document or folder presented in any of the views by clicking on its hotspot in the view (for example, it's text label or icon). To return to My Documents, you can use the Previous function, or link back via the Quick Sheet. If the document you link to is a subfolder of My Documents, then what you get is just like My Documents: a folder document arranged as I've already described.

IntelliView logic understands about navigating through folder hierarchies and helps you with Next/Previous navigation. When you link from a parent to a child, IntelliView adds the sequence of children in view sequence as a branch in the navigation context. Further, unless overruled by a stronger heuristic, this becomes the default branch for Next. Take mail for example. When you link from the inbox to a mail message, successively tapping Next will walk you from message to message. When you return to the parent, the branch of children is removed from the navigation context.

I believe the above combination of facilities eliminates the need for side-by-side viewing panes, though of course the PV can provide those kinds of views too. That works best on a two-display PV, where the hierarchy pane is wired to the left screen, leaving the right screen free for content. In this mode, the PV acts like a single-screen model, with the right display serving as the single screen (i.e., this is the screen that the corner buttons would act on). So, content on the right is still presented in a scroll-less, page-oriented format as normal. However you need to use scroll bars to move through the hierarchy list on the left. (This avoids any notion of "focus" in deciding which of the independent screens navigation controls apply to.)

Split-pane views on a single-screen PV works as you expect, by dividing the screen and then acting as I described for the two-screen case.

Now a technical aside. Given that all "documents" I've mentioned—My Documents, Mail, Favorites, etc.—are really folders with children, you might be wondering where the information comes from that causes the folder to presented in the document-like form I've defined. Why doesn't it just look like a folder?

To answer this, recall from my Storage Unification memo that in integrated storage, folders and documents are the same basic thing: each can have a content stream as well as children. So, for a folder, the data defining its document presentation exists in the folder's content stream.

To implement the organization I've described in this section, the folder's content stream has HTML text that defines the table of contents, the sequence of sections, and within each section, the view object that's to be presented. Obviously, you could customize or even replace this template any way you choose, for any or all of the folders I've said it applies to.

6.4 Note Taking

Note taking is the activity that, with today's devices, gets no respect. Note taking as a computer supported activity is generally so cumbersome and inconvenient that people just keep notes on paper. Users even still jot notes on paper while at their desks, in front of their PC's. (I certainly do.) This is a shame considering how much better PC's are at finding and presenting information once it's been captured.

For electronic note taking to be useful, several criteria must be met:

- Ability to directly annotate the material on which notes are being taken. For textual and graphic material, this means being able to jot notes directly on that material. For streaming media, it means being able to jot notes in context with the stream, so that the notes and media stream can refer to each other for retrieval purposes. Annotations on separate note sheets also need to be supported.
- Full complement of "pencil and paper" note taking functions, including text, drawings, erasure, and highlighting.
- Minimum, modeless note-taking UI. Taking notes needs to be smoothly integrated with the reading activity, or in the case of streaming media and realtime meetings, with the listening/viewing activity. This means that note taking must not involve UI manipulations like menu/toolbar actions to initiate. Any such manipulations distract the user's attention from capturing the note, an especial problem in realtime meeting situations, where any distraction will take concentration from the listening and note taking activity.

As you can see, no current note taking technology except pencil and paper comes close to fully meeting these criteria. For notebook and desktop computers, the problem is mainly lack of an invisible enough UI, and dependence on a keyboard and pointer as UI implements. (While powerful for entering and especially editing lots of text, the keyboard and pointer are far less transparent and efficient to use than a pen in the typical note-taking scenario.) For PDA devices, the problem is both UI and poor screen size and resolution.

The PV solves these problems by providing a sufficient display and the right UI for the note taking task. Beyond this, the PV smoothly integrates text and voice note taking in a way that makes the combination convenient and synergistic. It is the first device to achieve rich information capture (see my New Technology Areas memo for more discussion on rich info capture).

To get the fullest complement of note-taking functions, you need to use a pen. The most sophisticated PV's will come with a pen device that resembles a ballpoint pen. The pen has a writing tip that extends and retracts with a twist of the barrel. Extend the tip and the pen writes ink. Retract the tip and the pen acts as a highlighter. Write with the other end of the pen and the pen erases (like a pencil eraser). The PV distinguishes how the pen is being used by sensing the surface area in contact with the touch surface, or else by differences in electrical properties like capacitance.

You can also use your finger if all you want to do is highlighting, erasing, or clipping operations. This is actually useful, since often when reading all you want to do is add or remove highlights or save clippings for later reference. You can also use the finger for selection and transfer operations, like deleting and filing mail messages.

6.4.1 Highlighting

The PV lets you highlight material on a page using all the same conventions you have available on paper. To highlight text or a figure, just swipe it with the highlighting tip of the pen (writing tip retracted), or else with your finger. There is no mode to enter—just swipe. The effect is exactly the same as if you were using a real highlighter, and you can achieve all the same effects: highlighting the entire background of a stream of text or of a figure, or just highlighting a line down the margin, around a figure, or whatever. Preference settings let you choose the current color of the highlighter.

The PV also supports a few highlighting shorthands for highlighting large areas.

- To highlight an enclosed shape like a box or circle, draw the shape's outline with the highlighter, but don't lift the highlighter when finished. After a short delay, the shape will auto fill with highlighting.
- To highlight a run of text, swipe the first part of what you want to select, then hold. After a short hold interval, the selection will auto-extend successively as long as you hold: to end of word, sentence/line, paragraph, page, section, and finally document. An indicator in the command shortcut bar gives feedback on the extent of the selection (see Section 6.5.1.1). In tables, the highlight extends to end of row and then by rows, or to bottom of column and then by columns, depending on the direction of the swipe.
- You can also highlight a run of text lines by drawing a highlight line down either margin, next to the desired lines, and holding. After a short delay, the adjacent lines will auto-fill with highlighting.
- To highlight an entire page, highlight the page number and hold.

Highlights can be erased. To do this, either swipe with the erasing tip of the pen, or use the erasing gesture with your finger. The erasing gesture is a swiping motion where you jiggle as you swipe (the same motion you use with a pencil eraser). The shortcuts defined above for highlighting runs and areas work for erasing too.

Whenever you highlight, an implicit anchor to the highlighted material is established. This lets the PV maintain the proper highlighting in cases that the display needs to be repaginated.

All highlighting operations you do are also indexed by the PV with reference to the current time of day/date. This comes in handy for retrieval of associated voice notes (Section 6.4.6).

6.4.2 Jotting

Jotting (ink notes) works much like highlighting. With the writing tip of the pen, simply write on the page as desired. You can jot text notes as unrecognized printing or handwriting anywhere on the page, and you can likewise draw graphics. Advanced PV's may autorecognize printed text, but I won't delve into that option in this document (I don't believe it's useful for note taking until accuracy is well over 99%).

To erase your jots, use the pen's erasing tip, or use the erasing gesture with your finger. This works the same as was described in the last section for erasing highlights.

If you've highlighted a jot you've made, then erasing will first erase the highlight. You can erase again to erase the jot. Alternatively, you can erase a box around the jot and hold; both the jot and any highlights will be erased at once. The other shortcuts described in the last section for highlighting and erasing runs of text and areas will work for jots too.

If you use the pen to encircle or underline text an implicit anchor is established so that the ink associated with the text will stay associated with it if the PV performs any kind of repagination operation (see below).

Unlike for applying highlights, you can't use your fingertip to make jots. You can however use the fingertip to erase jots.

The PV has a trick that makes jotting on a PV actually better than the paper equivalent: if you jot anything besides underlining between lines of text, the PV will open more space for your jotting. This may cause repagination if lines must overflow the page. Depending on the document being read and on preference settings, the repagination may flow to the end of the document or just cause text to spill over onto a newly inserted page (e.g., page 27 flows to a new page 27a inserted into the page stream).

All jots are indexed by the PV to the current date/time so that rich retrieval of any associated voice notes can be done (see Section 6.4.6).

Besides jotting on a document's pages directly, you can also jot on blank pages at the beginning and/or end of every document. As described in Section 6.3.2, every document has a virtual first and last page just past the real first and last pages. One of the commands on these pages is labeled "Press for new blank page." As the name implies, this adds a page to the start or end of the document for you to take notes on. If the document is read-only and/or belongs to someone else, your new page is stored as an externally linked object in your own workspace.

6.4.3 Notes

Besides jotting directly on a page you can choose to capture notes on separate sheets, as opposed to writing on top of the material you're reading. To do this, you tap the desired insert point with the pen or a fingertip. A note icon appears.

Now hold the icon to open it. Alternatively, you can hold the spot where you want the note to appear. A context menu will appear, from which you can pick the Insert verb to insert the note (see Section 6.5.2).

Exactly what you see when a note opens depends on what model of PV you have. On a dual-display PV, the document page you're annotating shifts to the left display if it's not already there and the note sheet opens on the right display. On a single display PV, the note sheet simply opens up on the single display. In either case, you can now write your notes on the sheet.

Now, the note isn't merely a little sheet like a Post-it, it's actually a document. More precisely, a note is a document that's created and linked into the document you're annotating. (External links are used to do this so that you can take notes on read only documents, and share notes with other people.).

So the note icon is just a link icon, and when you open a note you are linking to it. Because notes are documents, you can:

- Turn the page of the note sheet if you need more pages of notes. The note document can append new pages to itself as any document can (see last Section).
- Highlight and erase your notes the same way that you highlight and erase on any document page.
- Insert notes on your note sheets. That is, notes can contain notes. (Of course, since this is just adding links.)
- Use the Previous button to return from your note to the document you were annotating. (Or, as for any document, you can go to the Quick Sheet and link from there to wherever you want).

Note that when creating a note there's nothing to close or save. When done with it you just navigate away. If you want to delete it, just erase its link, or choose Delete from the context menu (Section 6.5.1.2). The note form also includes a delete button that you can use to delete the note while viewing it.

As a further aid to using notes, the note documents support a title field on page 1 of the note. Anything you enter in this field will appear in a Notes Index for the document the note is attached to. For example, if you're adding a note to page 27 of War and Peace, and you title the note "Plot question", then the Notes Index for the book will show an entry like this:

```
Notes Index
    :
    Plot question....................27
    :
```

Where "Plot question" is the ink (or recognized text) you scrawled.

You can get to the Notes Index either by paging to the end matter of the document in question (where the Notes Index is placed), or by navigating to the document's table of contents and linking from there. A Notes Index TOC entry is created the first time you insert a note in the document and removed when you delete the last note.

You can also review notes by visiting the Notes Folder. This is a view on the annotations folder (Section 6.6) that shows all notes you've made in all documents. The default view categorizes the notes by document, and under each, shows index entries as for the Notes Index. By default the documents are listed from most recently read to least, but you can change the sort order if you like via view options.

So far I've described notes as items that are associated with a particular spot in the text. However, if you wish, you can also indicate that the note is associated with one of several ranges associated with the point where it was inserted:

- The insert point only (no associated text). This is the default.
- The current selection (see Section 6.5.1.1).
- The highlighted text that contains the note.
- The page that contains the note.
- The section/chapter/document that contains the note.

You can change this setting at any time. The graphic for the note icon changes depending on the extent of the note, so that you have a visual cue for what range it's anchored to.

The main use of this ability to specify a range for a note is in collaboration scenarios, where you send notes to other people (Section 6.9). The range setting lets the recipient know exactly what the note pertains to. And, in situations where the recipient isn't known to have access to the original document, the range setting tells the PV how much text to attach to the sent note. (If the recipient does have document access, for example because they're in the same workgroup, then the note is sent only as an external annotation without a copy of the anchor range).

Notes are the basis for two other PV features: bookmarks (Section 6.3.3) and clippings (Section 6.4.4). These are the same thing as a note, except the link icon for them wants to appear other than in the body of the text document itself. A bookmark wants to appear as a tab in the margin of the document containing the bookmark, with an optional bookmark icon in the text body to indicate exactly what spot was bookmarked. A clipping wants to appear as an item in the Clippings Folder, as well as a clipping icon at the clip point (as a reminder that this material has been clipped).

The distinction between inserting bookmarks, notes, and clippings is normally made based on where you the display you tap; e.g., tapping in the bookmark margin inserts a bookmark, in the display context, a note. But you can change this setting on the note form if you want. Three checkboxes let you indicate whether the note should appear within the page (note icon), in the bookmark margin (bookmark tab), and/or in the Clippings Folder (clipping icon on margin). Any combination is valid.

You can change the note/bookmark/clipping settings at any time. So for example, having created something as a note, you're not married to that choice and can turn it into a bookmark and vice versa if you want.

Notes have one other feature that is useful for collaboration scenarios: a send button. This is explained in Section 6.9.

6.4.4 Clippings and the Clippings Folder

A common activity that's related to note taking is the ability to "clip" material for later references. For example, you may want to clip an interesting news story, a chapter or section of a book or document, or a few useful paragraphs from a magazine article. Therefore, you need ways to easily indicate what material to clip, to go back later and review the stuff you've clipped, and to file it or send it to other users as appropriate.

Clipping is done by tapping in the clipping margin (default: left margin) next to the material to be clipped; or you can do this via the context menu Insert verb (Section 6.5.2). This inserts a clipping icon and makes an entry in the Clippings Folder. (The actual clipping is stored as an external annotation in the annotations folder). The clipping logic uses a simple contextual heuristic, described below, to decide how much material you probably want to clip: page, section/chapter/article, or highlighted text. The amount clipped is encoded in the graphic for the clipping (for example, using different embellishments to distinguish page versus article versus range of highlighting).

You can add notes to a clipping or change the clipping settings by opening the clipping form. You do this by holding the clip. The clipping settings are also available as properties.

As mentioned in the previous section, clippings are actually just a kind of note. In particular, they are notes that have the "clip" option box set and that are associated with some non-null range of text. The range options are the same as the anchor range for notes (see last section) and can be set the same way on the clip's note form.

The clip's range setting defaults automatically based on context in which you made the clip. If next to a bunch of highlighted or selected text, the highlighted or selected range option is defaulted. If next to the title of a document, chapter, or section, then the corresponding document/chapter/section option is defaulted. If on a page not next to highlighted text or a title, then the page option is defaulted. You can change the range setting on the clipping form if you want.

Since clippings are notes, you can of course use all the features of the note form, including jotting or recording notes to yourself about the clip, or sending the clipping to another person or to an arbitrary file folder. You can also change the settings of the clip, note, and bookmark checkboxes as desired to turn the clipping into one of those other entities or even for example to make the thing appear as both a clipping and a bookmark.

When you're ready to review your clippings, you can go to the Clippings Folder. You can do this via the Quick Sheet or My Documents, or you can add the Clippings Folder to Favorites to access it that way.

The Clippings Folder carries all clippings you've made, not just clippings in a particular document. View options let you organize this list chronologically as well as categorized by document and by topic (based on topic keywords entered into the clipping form); or you can make up your own views as explained in Section 6.3.8.

The canned views on the Clippings Folder show you the document and page number where the clipping came from and the date/time it was made. The views also show you the title field of the clipping (if you filled that in) and the text in the clipped document that is most adjacent to the clip. Tapping the clipping title opens the clipping form for this clip. Tapping the document text field or page number takes you directly to that point in the original document.

6.4.5 The Notebook

For note-taking convenience the PV includes a built-in notebook. This is just a blank document for the user to write and/or dictate notes in that aren't associated with a particular document. You can link to the Notebook from the Quick Sheet or from My Documents, or you can put it into Favorites for access that way.

6.4.6 The IntelliTape Recorder

Sometimes it's more convenient to dictate notes than it is to jot them, or sometimes you may want to do both. On PV's equipped with voice input you can do this.

Many newer PDAs offer basic voice memo functions. They're of limited use, because voice memos on these devices don't interrelate to reading matter or written notes. And lacking wireless links and streaming audio, their audio storage is limited by memory capacity.

The PV's approach to this is to go much further than any existing device. The goal is to work like a cassette recorder, but with functions that richly integrate voice and textual material, and which exploit wireless streaming audio when the situation permits. The PV's combination of advanced recording features and heuristics is called IntelliTape.

Using the IntelliTape recorder you can:

- Record, play back, and scan dictations exactly as you would on a cassette recorder.
- Edit your recordings, adding to as well as inserting and deleting within previously recorded dictations.
- Insert dictations into specific spots on document pages, where they appear as notes.
- Automate voice capture by engaging an intelligent voice-activated recording feature.
- Instantly start playback of whatever was recorded when a given page was being viewed.
- Instantly start playback of whatever was being recorded when a given ink note was written or when a given highlight was being drawn.
- Synchronize audio playback to page viewing so that as a recording plays the pages advance in synchrony, and so that as pages are navigated, the recording seeks in concert.

6.4.6.1 Recording Model

The recording model is basically that of a cassette recorder. You can think of the PV has having a single, long "tape" that you can record on while viewing documents. Pressing play will start playback at the current tape position, and pressing record will start recording at that position. An indicator that's part of the recorder controls gives you graphical and numerical feedback on your position in the tape. Standard fast forward and rewind controls let you scan around in the tape, and you can record over stuff you recorded earlier.

To begin recording and listening to dictations, a naïve user doesn't need to understand anything more than that much, which is nothing more than knowing how to use a cassette recorder. The PV's basic controls emulate this exactly.

Now, acting like a cassette recorder may make the PV easy to pick up and use if you're a naïve user. But for anyone more experienced, particularly a heavy user of verbal note taking and collaboration, the model would be inconvenient and limiting. Who wants to hunt around in big tape (or a library smaller ones) for material you recorded days ago about a particular document? Who wants to suffer with a cassette recorder's limited editing ability?

So, although the PV emulates a cassette tape model, it actually uses a more sophisticated model under the hoods—a model that lets it offer richer features to the more experience user while still acting like a cassette recorder.

To make this possible, the PV's "tape" isn't just a linear file of audio bits; it is a database of independent sound clips. Every time the PV starts recording, it makes a new clip out of whatever it captures from that point until the point where recording stops. (When using voice activated recording, each interval captured becomes a separate clip.)

Clips have properties that the PV keeps track of, like the document in view when the clip was recorded, and what the document position, relative tape counter reading, and actual date/time were. This means that the PV can index into and manipulate recordings in multiple ways.

For example, the PV can assemble all the clips that belong to a particular document in relative tape counter order, thus creating the illusion of a cassette of just that document's comments. Or it can assemble clips sorted by document within a given date, thus creating a cassette of all the notes a user recorded on a given day. It can go from a tape counter value to a document position and vice versa. It can maintain multiple audio streams for a document (the document "cassette" plus other, embedded cassettes). And so on. This is all just a matter of defining views on the clips database.

The richness this model permits will become clearer as I go through the recorder's functions. Yet the thing to remember is: for the utterly naïve user who only uses the normal cassette controls, it all works exactly like a cassette recorder, with one long, permanently loaded cassette (called the "master tape").

6.4.6.2 Recorder Controls

The recorder controls are normally hidden but can be made to appear at any time. They are implemented as a toolbar (Section 6.7), which by default is hosted in the command shortcut margin and initially closed. This means you'll see a toolbar tab for the recorder in the shortcut margin (analogous to a bookmark tab). Tapping the tab opens the recorder controls into the margin. Visually, the toolbar slides out from the margin edge. Tapping the tab again retracts it, leaving only the tab.

Like any object, the recorder toolbar can be deleted or moved as you please (e.g., just erase over its tab, or select over it and delete or move). Even if you delete it, you can always grab another copy from the Operating Manual.

The PV comes with two versions of the recorder toolbar, one that contains a basic control set, the other that contains the complete (advanced) control set. You can also make a custom control set using the normal procedure for customizing toolbars (Section 6.7).

In this section I'll enumerate what's in the two control sets. The sections that follow explain their operation in detail.

The basic control set includes:

- Play, Record, Stop, Pause, Fast Forward, and Rewind. These work exactly like those on a cassette recorder. The Record button has a light that comes on when recording is engaged. The light flashes when data is actually being captured (it is not flashing in record mode when only silence is detected). One embellishment for playback: pressing play while already in playback or record mode will automatically start playback at the point where playback or recording started (i.e., repeat what you just heard or dictated).
- Index Back and Index Forward. Analogous to those on a CD player, though here they move the tape position back and forth between clips in the recording.
- Indicator. Resembling a cassette tape, this shows the tape position both graphically and via a tape counter below the icon. An additional graphical embellishment indicates when the tape is positioned just before the first piece of material recorded with respect to the current page, or just after the last piece of material for that page. The tape indicator flashes when playback or recording is in progress.
- Tape. Pressing this button toggles the button label between "Master Tape" and "Document Tape". When in Document Tape mode, the recorder will automatically "load" that subset of the master tape that pertains to the document being viewed. When off, the PV master tape is loaded.
- Basic/Advanced. This button toggles between the basic and advanced control sets.

The advanced control set includes all the basic controls plus:

- Display. Pressing this cycles the tape indicator between showing the tape position as a count, as a percent of the total recording, and as duration from the beginning of the tape.
- Edit. Pressing this cycles the Record button from being labeled Record, Insert, and Delete. This affects whether engaging record will overwrite, insert, or delete recorded material.

- AutoRecord. When this toggle is on, the recorder employs smart voice activation logic to engage recording when a nearby voice is detected. When off, any ambient sound above the silence threshold will engage recording. An indicator light on the toggle indicates when it is on.
- AutoSeek. When this toggled is on, the recorder automatically "seeks" the tape position to keep it coordinated with the current document position. Likewise, it will navigate the document when the user manually seeks the tape. When off, the document and tape positions are independent.
- Search. Opens the audio search form, from which voice searches are initiated.

6.4.6.3 Basic Recording and Playback

You do basic recording via the basic recorder control set, or that subset of the advanced controls. Usually only naïve users, or those very married to cassette tapes, will prefer to use the basic controls set.

As I described previously, the user's basic model is that of a single, large cassette tape. The controls work as you'd expect, even down to pressing play and record simultaneously to engage record mode. As a convenience for dictation, you can also simply press and hold the record button to start recording. Releasing the button stops recording.

The Fast Forward and Rewind buttons lock down when you press them, as they do on a cassette recorder. You use the Stop button to stop the tape. A Settings Sheet option lets you change this behavior so that the buttons don't lock down. In this case, the tape continues to scan only as long as you hold the button down; no need to use the Stop button.

To adjust the speed of Fast Forward or Rewind, you can slide a finger along the left or right edge of the PV, much as you do to control page riffling speed.

Play and Fast Forward or Rewind can be engaged at the same time. In that case the PV uses a compression algorithm to play back an excerpted version of the audio stream as the tape winds. This gives you audio cues about where the tape is positioned.

The Play button also has a feature that's handy for dictation scenarios. If you press Play while the recorder is already playing or recording, the tape backs up and plays from the point where playback or recording was started. In other words, with a single button press you can repeat what you just listened to or recorded. Playback or recording resumes after the repeated interval.

A drawback of the single-tape model experienced with the basic controls is that it makes it hard to go back and find material pertinent to the document you're currently viewing, let alone the current page. This is more than just a problem of seeking to the right place in a long tape. Because the tape is linear but a user's navigation experience among a set of documents probably is not, the set of dictations pertinent to a given document may be scattered throughout the tape.

Such are the limitations of traditional recorders...but not of the IntelliTape recorder.

To solve this problem, the IntelliTape recorder supports the Document Tape feature. When engaged, the recorder automatically "loads" only those clips associated with the current document. It's as if a sound engineer culled the relevant clips and assembled them into the right sequence for you on a tape of their own—the document tape. (In fact, the recorder is simply querying the database of clips for a match on document name, sorted by tape counter).

The automatic loading operation happens every time you switch documents. The tape is positioned at the first clip, in tape counter order, for this document.

You're free to scan, play, and record on document tapes as for the master tape. Recording at the end of the tape extends it as it does for the master tape, so the user model for the two kinds of tape is the same. (At the level of the master tape, the material being appended to a document tape is in effect being inserted into the master tape stream.)

If you wish, you can use the Document Tape feature to create a library of individual cassette tapes, each with its own title. Each "cassette" is a document containing only an audio recording. To listen to one of these you open the document and press Play. The real power, of course, is when you use the Document Tape feature as a way to collect and review voice annotations in documents that contain text and other information.

As a further navigational aid, the recorder has Index Forward and Index Back buttons that work much as on a CD player, except here the "tracks" are the clips in the audio stream. The buttons work like this:

- Tap Index Back. Seeks to the previous clip in the tape
- Tap Index Forward. Seeks to the next clip in the tape
- Hold Index Back. Seeks to the start of the first clip associated with the current page being viewed. (This is what AutoSeek does for you when it's enabled.)
- Hold Index Forward. As for Index Back, but seeks to the end of the last clip for the current page. (This is mainly useful with the advanced control set, where recording can be made to insert rather than overwrite—you'd seek to the end of the page and insert more comments for it).

You can use the index buttons when play mode is engaged. The recorder will seek and resume playing at the new point.

To make it obvious when a given page has comments that were recorded when that page was in view, a cassette icon appears next to the page number. Tapping this icon seeks to the start of the first clip associated with that page.

Unlike CD tracks, clips of course aren't on boundaries that the user will remember or be able to predict. The boundaries depend entirely on how tape was recorded and edited. Nevertheless, it's most common for each clip to contain a complete utterance, so navigation by clip should be a useful complement to the Fast Forward/Rewind buttons, especially when editing recordings.

The rest of the recorder functions I describe in the following sections are available via the advanced control set.

6.4.6.4 AutoSeek and AutoRecord

When using the recorder, it's most common to want to capture and review notes with reference to the material you're looking at right now. You want to access recordings in context, such as whatever comments you spoke while reading a given page, or while jotting a given note or highlighting a given area. And you want to be able to access and make new recordings immediately, without mucking with cumbersome UI operations.

The IntelliTape recorder makes this possible via two features called AutoSeek and AutoRecord. You can toggle each of these features on and off via the advanced recorder controls.

AutoSeek causes the recorder to automatically synchronize audio and document positions. So, with AutoSeek engaged:

- If you navigate to a new page and press play (or are already in play mode), AutoSeek will start playback at the first clip for the new page. Tapping the cassette icon adjacent to a page number will restart playback with the first clip for that page.
- While viewing a page, if you tap an existing ink jot or highlighted area, AutoSeek will start playback at the first clip that was recorded when that jot or highlight was made.
- If you navigate to a new page and press record (or are already in record mode), AutoSeek will start recording at the end of the last clip for the new page (i.e., new comments are inserted after existing comments).
- If you navigate the tape while playing using the Fast Forward, Rewind, or Index buttons, or if you just let the tape play, AutoSeek will navigate the document to keep place with the recording.

AutoSeek eliminates the need to manually navigate the tape in most cases. You can simply turn to any page and start listening to the comments for that page, or add new comments to the page, all without manually positioning the tape. You can likewise hear comments associated with any ink jot or highlight just by tapping it. Of course, you can still position the tape if you want to edit or scan the previously recorded comments.

Much as AutoSeek mostly eliminates the need to manually navigate the tape, AutoRecord eliminates the need to manually start and stop recording. AutoRecord implements an advanced kind of voice-activated recording that only captures sound when it determines that a nearby voice is speaking; in other words, extraneous ambient sounds won't trigger recording. Furthermore, AutoRecord manages the Record button. Any time the tape is not being played or seeked, it will engage recording if it detects your voice.

When AutoRecord is off, recording only occurs when you manually engage recording, and all ambient sound is recorded. (However, long silent intervals are compressed out).

AutoRecord leaves you free to read though a document, making comments as you go, and never touching any recorder buttons. Turn the page, make comments, visit another document, make comments, etc. Even play back stuff in between making comments, with no need to press Record again to enable recording. In situations where the PV can stream audio out the wireless or IR ports, you can leave it in AutoRecord mode virtually all the time, since stuff is only captured when you speak.

AutoRecord works using a combination of loudness, spectral, and possibly rhythmic characteristics to distinguish a very nearby voice from background noises, silence, or more distant voices. In an advanced implementation, the PV could use speaker-dependent recognition to truly cue itself only on *your* voice.

While I think people will use AutoRecord most of the time, there is one scenario where it's better to turn it off: when you want to record a transcript of a meeting. Here, you do want to capture all the ambient sounds, not just your own voice. A particularly handy thing about making a meeting recording is that you can later go back and review it in concert with your written notes. With AutoSeek on, just visit a page of the meeting presentation to hear what was being said at that time, or tap any of your notes to hear what was being said when you wrote it.

6.4.6.5 Editing Recordings

Because it emulates a cassette recorder, the default behavior of the PV is to overwrite when recording. That is, starting from the current tape position, the recorder will overwrite the sequence of previously recorded clips.

For the technically minded, some amplification: If you record over part of an existing clip, that clip is truncated and your current recording is a new clip. If you record over the entirety of an existing clip, that clip is deleted. The user never needs to understand any of this, since the result is to emulate a linear recording medium for the basic play, record, and wind functions.

Overwriting is fine if you're a naïve user or are really used to using dictation machines, but it's pretty crummy if you need to do any real editing. For this you need insert and delete functions.

The advanced recorder controls include an Edit button that affects the behavior of the Record button. Pressing Edit cycles the label on the Record button among Record, Insert, and Delete. Depending on what the label reads, engaging the button will cause newly captured sound to be overwritten or inserted at the current tape position, or it will cause stuff to be deleted from the current position. So that you will know what you are deleting, engaging Delete will play back material as it is being deleted. It may make sense to mix in a noticeable background tone or sound effect as a cue that what you're hearing is being deleted. You can use the index buttons while deleting to automatically delete forward and back in sound-clip increments, as well as to the beginning or end of the current page's comments.

6.4.6.6 Embedded Recordings

Sometimes it's convenient to associate recordings with specific spots in the text of a page. You can do this by opening a note and making your recording, just as you would for any document. Think of the note as if it were a cassette tape embedded in the document. Just as you can open a text note to read the notes, you can open a voice note to play it back. Of course, your note can contain both text and a recording if you like.

Besides letting you attach voice comments to specific spots in a document, using notes to embed recordings has the plus that you can give your recordings a title, using the title field of the note form. This means that your recordings will show up in the Notes Index, and it will be much easier to find and review your recordings at a later time.

Because creating and listening to embedded recordings will be a common scenario, the note form has some logic to make this nicer by making the playback and recording of embedded notes one-touch operations. These features are called OneTouch Recording and OneTouch Playback.

First, to make it clear which notes contain recordings, the note icon changes to reflect whether the note contains text only, sound only, or both. For example, if only a recording, it will look like a cassette tape. When you look at a page you can see at a glance what kind of notes you've embedded where.

OneTouch Playback lets you simply tap a note to initiate playback; no need to manually open the note and press Play. (For OneTouch Playback to work, Document Tape must be engaged.) While the note plays back, its icon will flash, and you can use the recorder controls to control it. When playback comes to the end, or you press Stop, the recorder returns to the containing document's recording context. You can also stop playback of a note by tapping its icon again, or you can tap another note to start playback of its recording.

OneTouch Recording is analogous to OneTouch Playback. With AutoRecord active, just tap the spot where you want the new recording to be inserted. A note will appear, flashing, and will record whatever you say. To finish the recording you can:

- Tap the note to return to the document recording context
- Tap somewhere else to create another new note, which the recording context will switch to
- Tap another existing note to switch recording to that context

The combination of OneTouch Playback and Recording means you can go through a whole document listening to and adding recordings without ever using the recorder controls. As you read, simply tap existing notes to hear them, and tap in blank space to create new notes that you can record into. With AutoRecord active there's no need to manually switch between record and play modes, either. The recorder just does the right thing.

6.4.6.7 Searching

An advanced implementation of PV's may support voice search. This function lets you dictate one or more search terms that the PV will use to search your voice notes. There are two ways that this function can be supported:

- When you record dictations, you can deliberately enunciate important words in a separated manner (that is, bracketed by short silences and possibly emphasized in volume). The PV can recognize utterances with these characteristics as probably being search keywords and keep track of them accordingly.
- Dictations that are uploaded to servers can be processed by more sophisticated continuous speech engines.

Here then is how speech search would work. Pressing the recorder's Search button opens a search form. To initiate a search while on this form, you dictate your search terms as separated speech. You may optionally use search fields to scope the search according to date/time, document, and page ranges.

The PV will proceed to search for the desired keywords using a matching algorithm to compare the search terms versus previously stored voice notes. It may do this internally if it has stored audio notes that contain separated words, or by shipping the request out to a server if your voice notes are server-based. The server would employ a much more sophisticated search engine that may even be able to find words in continuous speech streams.

Note that servers with sophisticated speech processing engines may also be able to support text search of speech streams, a feature that would be useful for desktop users. The way this would work is to process the speech stream once when it is first stored, looking for "significant" keywords the same way that a full-text indexer does. These could be given to the full-text indexer as if the speech file were a text file, so that speech clips could later be retrieved based on these textual keyword queries.

6.4.7 Linking

PV users are not limited to the links that are authored into content; they can create their own. The PV implements the rich linking model I've written about in my prior Storage Unification and Hyperactive Desktop memos. This means that links can have source and destination anchor ranges as well as their own properties. Despite the richness of available features, users can create links as easily as they can copy text, and all with a fingertip (the pen isn't required).

Links, by default, manifest themselves as a change in visual attributes associated with something that already exists in the text. For example, a user who wanted to put a link in this paragraph, for example on the word foo, does so by associating the link with that word, rather than by inserting a separate object to represent the link. When the link is created, the visual attributes of the source anchor change to reflect the link's presence.

This is an important point. From the user perspective, the default way to use links is to select some existing thing to act as the link hotspot, and then to designate where the link leads to. Nothing gets "inserted" into the text. (And in fact, user-created links are stored as external links, so you can create links in non-editable material).

A user who wants to insert something new to act as the link hotspot can of course do so—for example, insert a jotted word or even a note and then link it to someplace else, so that tapping the jot or the note icon will take you to that other place.

The PV lets users create and delete links as well as view and manipulate their properties. These operations are all part of a general family of editing operations, the UI for which is discussed in Section 6.5. Here I'll just cover the link-specific properties. Note that looking at link properties is an expert user feature; even users who create their own links wouldn't need to look at link properties unless they're trying to do something fancy.

To get at a link's properties or to perform other operations on it, you select its hotspot and hold the selection. Selection mechanics are explained in Section 6.5.1. The result is to open a context menu that, among other things, includes a set of link verbs:

- Link Properties. Opens the link property sheet.
- Delete Link. Deletes the link without affecting the underlying content.
- Disable (Enable) Link. This toggles the link behavior on and off (see below and Section 6.5.1.2). The link's anchor range is given a muted emphasis to indicate that a link is still present but disabled.

The link property sheet is implemented as a document much like the note form. It has several pages. The first page is the subset of properties that users would most commonly change, presented in a way that less expert users can understand. Subsequence pages provide the full set of advanced properties and options. Only the most expert user (a content author) would ever muck with these.

For example, the first page may contain just the property comments (a notes field), information about where the link leads to, and a chooser that lets you choose from a few options on how the hotspot should appear (e.g., emphasized text, button, icon, or thumbnail).

The subsequent pages give full access to properties including the source and destination anchor specs, the visual extent and appearance of the hotspot, and the link behavior options. Each page of the link property sheet is bookmarked so you can jump directly to it if you want (analogous to property sheet dialog tabs, but here done without a special UI mechanism).

Together, the set of link property pages encompasses a lot of information, including general properties, source and destination anchors, hotspot characteristics, and link behavior. Let me reiterate that only an expert content author would ever look at this stuff.

- General properties. These include the type and file system properties of the link. Link type information describes the semantic nature of the link and how it relates the stuff being linked. Only authoring and viewing software would ever access the type properties, as they pertain to the organization of the material containing the links. Type tells you whether the link expresses a parent, child, or peer relationship, and whether the destination represents:
  - A document component: a figure, table, footnote, or other cross-reference.
  - The next or previous "page" (in the web sense) of the current document/topic.

  - The next or previous document/topic in an authored web of documents/topics.
  - Something not part of the current document/topic: a comment or hyperlink.
- Source anchor. Lets you choose whether the link is anchored to a character, word, paragraph, image, part of an image, table row, cell, or column, or an arbitrary range of document positions. This parameter is automatically set according to the source selection the user makes when creating the link (Section 6.5.1); the property gives you a way to change the anchor if you need to.
- Hotspot. Lets you specify the physical extent and appearance of the hotspot. By default the extent matches that of the anchor, but it can be made bigger or smaller. For hotspots on text, the default appearance is blue underlined text, but the foreground and background colors and text attributes can be changed. Other appearance options include manifesting the link as an icon, button, thumbnail of the link target, or as an in-place (active) rendering of the link target. Hotspots can also be invisible, which is appropriate for links over GIF images. Another hotspot option lets you specify how the link is previewed: any or all of the name of the target, a thumbnail of it, and specific balloon text.
- Destination anchor. Let's you choose the target of the link and specify its range. The basic link creation UI results in destination anchors that are a single document position (a "point") rather than a range. Destination ranges that aren't points are a very advanced feature mainly used in implementing certain viewing and collaboration features. For example, if the destination anchor is a range, the viewing software can automatically synthesize appropriate link preview information based on the content of the range.
- Behavior. Let's you specify the action to take on tap and hold operations. Choices include
  - Navigate. Takes you to the link destination.
  - Preview. Pops up navigational preview information as explained above.
  - Run. Causes the destination content to be executed, with specified run parameters. The target must be a command or script.

  Additionally, for links set to appear as thumbnails or in-place renderings, other behavior properties let you make the link hot, warm, or cold, and set the pre-fetch, refresh, and caching parameters for hot and warm links.

For more information on link properties and semantics, see my prior memos: Storage Unification and Webification, and Hyperactive Desktop.

To delete a link, you use the Delete Link verb on the context menu, or you can erase as you would do for a jot or highlight. For example, if you have previously created a link in a read-only document on the word foo, then erasing over that word will erase the link.

When a link exists over a jot or highlighted area, erasing follows a precedence order: highlights first, jots next, links last. That is, the link is only erased when the last of its anchor is erased. The same rule applies when erasing links in editable documents. As a shorthand for doing up to three erasures to remove a highlighted word or jot with a link on it, you can erase a box around it and hold. This will do all three erasures in a single operation.

Note that links are not restricted to just existing in the content area of the display. They can exist anywhere. You could, for example, place a link over a bookmark; the link would take precedence for tap operations, meaning that the bookmark now acts as a document-associated shortcut to some other document (whereas a bookmark is normally a link within the current document).

6.4.8 Controlling the Display of Notes

The preceding sections have talked about a lot of note taking activities: highlighting, jotting, annotating, recording, and linking. Having added lots of this material to a document, there are times when you want to turn it off, so that you can view or print the document "clean". Or, there are times when you want to view notes created in a different context from your personal ones, such as the shared notes captured during a meeting or other collaborative session (Section 6.9).

The PV includes a Notes command that lets you control what notes to display. It also lets you independently where notes that you create should be stored. The choices are:

- No notes. Notes won't be shown or recorded.
- Document. Notes are recorded in and displayed from the document itself (i.e., internal notes).
- Personal. Notes are recorded in and displayed from your personal notes (i.e., external notes in your Annotations Folder).
- Shared In.... Notes are recorded in and displayed from a specified shared (workgroup or meeting) folder (i.e., external notes in that folder).

A combination of choices is perfectly valid. Advanced options let you break down the control of notes sourcing based on the type of note: highlights, jots, and annotations.

Tapping the Notes command shortcut opens a sheet, the Notes Sheet, that presents the full set of options for the current document. You can also hold the Notes command to get a context menu that presents the basic choices as a checklist.

You'll find instances of the Notes in the Operating Manual and on the Current View Sheet (Section 6.3.8.1). On the latter the Notes command is configured to display as an in place object, meaning its settings controls appear within a subframe of the host sheet. (This is an example of the power the link model brings to things like commands; see Sections 6.1.1, 6.4.7 and 6.7.)

6.5 Editing

The PV is mainly a reading and note taking device, and most of the material you read on it won't be editable, or else the scenarios you use it for won't be editing scenarios. Nevertheless, advanced models of the PV and certain advanced scenarios do call for editing capability, so the design permits this.

As for other areas of functionality, the PV strives to support editing in a way that is easier and less UI-intrusive than current desktop approaches. At the same time, because the current desktop editing model is so well entrenched, the PV adopts an editing model that is a logical evolution of the desktop model. It is a model that would work well on the desktop.

There are really two levels of editing functionality, one of which will be more common on the PV than the other. The first, more common level is the performance of transfer operations, like copying, moving, deleting, and linking information that already exists.

For example, consider using a PV to read mail. You'll want to be able to file or delete messages as you read them, which means you need a way to perform the classic editing operations of selecting, moving/copying, and deleting. As for all other PV operations that are common enough, it is possible to do these operations with the fingertip.

The second level of editing is the full complement of text input and editing operations. These require the pen or the external keyboard/pointer accessory.

The editing model uses the familiar concepts of a current selection, edit cursor, and command verbs, with some improvements to streamline the editing process and to make transfer more flexible.

6.5.1 Transfer Operations

Transfer operations include the classic cut/paste and copy/move/link verbs. To perform one of these actions, you need to designate the source of the action, the verb, and (as appropriate to the verb) the destination of the action.

Like today's model, these steps are done by establishing a source selection, choosing a verb, and pointing to the destination.

Unlike today's model, you can do these steps in any of three different orders as you please:

- Verb first: Pick the verb, make the source selection, hold the destination
- Verb second: Make the selection, pick the verb, hold to the destination
- Verb last: Make the selection, hold to the destination, pick the verb

The verb first order is perhaps the most natural for naïve users who are initiating a transfer by finding commands in the Operating Manual. They find the verb they want and tap it, then are guided to make a source selection and to point at the destination. The verb second order is most familiar for current desktop users, while verb last (as you'll see) is the most direct. In fact, because it's so direct the feature has a name: OneTouch Transfer.

6.5.1.1 Making Selections

Selecting objects or text works pretty much as on the desktop, except here you swipe with the fingertip or a pen.

Now, the various swiping possibilities of the finger and pen have already been defined as being highlighting, jotting, and erasing operations. It's important to note that highlighting and selecting are *not* overloaded to be the same thing; that is, you don't draw a highlight in order to make a selection. You draw a selection instead, which is both visually and operationally a different thing.

The basic (naïve user) way to tell the PV that you are about to establish a selection is to choose the Select command. For convenience, this command is available on the Quick Sheet. In any editing scenario, such as when you initiate a selection, the PV will also automatically place a Select command shortcut in the shortcut area (right margin) as well, or you can place on there yourself (Section 6.7) to keep it always handy.

So, to make a selection, you tap the Select command, then swipe with the pen. You can use either the jotting or highlighting tip to do this—it doesn't matter. A Select command shortcut appears in the shortcut area and flashes or uses other visual emphasis to indicate that a selection is in progress. This is your visual cue that swiping with the pen will select rather than jot or highlight. Further, the shortcut's label gives feedback about what's been selected so far, as in "Selected: 2 paragraphs in My Letter" or "Selected: Revenue row in My Sheet".

As a shorthand for starting a selection, you can do a tap-swipe operation. That is, tap at the point where you want the selection to start and then swipe from that point in the direction you want to select.

To unselect part or all of a selection you've made, just erase, exactly as you'd erase a highlight or a jot. The area you erase over is deselected.

While selection is in progress, you're free to use the normal idioms for automatically extending your selection (or unselection) across runs of text and polygonal areas (Section 6.4.1). For example, you can

- Swipe over disjoint areas to make a disjoint selection
- Hold at the end of a selection to auto-extend successively across sentences, paragraphs, rows/columns, pages, and so on.
- Swipe down the left margin and hold to select entire lines or paragraphs
- Swipe a box shape or other enclosed shape and hold to select an enclosed area
- Erase down the left margin or in a closed shape to deselect lines, paragraphs, or areas

As you can see, selecting and deselecting works the same as drawing and erasing a highlight. The visual emphasis of a selection is of course different from that of a highlight, as is the meaning.

Note that unlike the desktop selection model, there are no selection modifiers, as in "ctrl-select" or "shift-select". You just use the pen, finger, or mouse to make or erase partial selections as needed, and optionally use the various auto-extend and fill idioms as shortcuts.

To cancel an entire selection without manually erasing the whole thing, just tap inside any existing part of the selection. Or you can tap the Select command that's present in the shortcut margin. (Tap and hold actions inside the selection versus on the Select command are always equivalent.) Tap the Select command again, or tap-swipe in the content, to start selecting again.

If you hold the inside of a selection, or on the Select command, you get a context menu that gives you choices to:

- Copy/Move/Link/Copy to Clipboard/Cut to Clipboard/Paste from Clipboard/Delete. Either one or all of these verbs will be present, depending on whether a transfer verb has already been chosen. See next section.
- Clear Selection. The entire selection is cleared. Select mode is still active.
- Continue Selecting. No operation—the menu closes and you can continue selecting.
- Cancel Selection. The selection is cleared and select mode is turned off. The Select command shortcut stops flashing. As a shorthand, you can also cancel a selection by tapping inside the selection.

6.5.1.2 Performing Transfers

Like today's desktop UI, the PV supports a clipboard model for transfers. It augments this with a direct transfer model (direct copy/move/link). The transfer verbs include Copy, Move, Copy to Clipboard, Cut to Clipboard, Paste from Clipboard, Link, and Delete. To streamline context menus, a Settings Sheet option lets you control whether to present only the clipboard or direct subsets of the transfer verbs, or both. Both sets of course are always available via the Operating Manual.

Of these commands, Cut and Delete operate only on a source selection and don't involve designating a destination. Paste on the other hand involves designating a destination only. The rest require that you both select a source and designate a destination.

As I said in the last section, you can choose what transfer operation to perform at different points in the transfer process: verb first (before selecting the source), second (after selecting the source), or last (after holding the destination). Verb last is also called OneTouch transfer because, after having made the source selection, you can complete the transfer with a single touch.

Here's an outline of how the different transfer sequences work:

1. Verb first.
   - Choose the desired command. It doesn't matter if where you find the command: in the Operating Manual, on the Quick Sheet, or on a context menu (hold on any spot that opens a context menu). Tapping a transfer command automatically puts the pen/finger into select mode (as if you'd tapped the Select command or done a tap-swipe). A shortcut for the transfer command is added to the shortcut area if one isn't already present; this is an instance of a select command shortcut (see last section), except labeled to indicate the type of transfer in progress; e.g., "Copying: 2 paragraphs in My Letter".
   - Make a source selection as described in the last section.
   - Hold on the spot where you want the transfer to complete. A context menu opens that includes a completion option for the transfer command you chose (e.g., Copy Here). A tracking cursor also appears for fine-tuning the destination if necessary (see below). Choose the completion command from the context menu to carry out the transfer.
2. Verb second.
   - Make a source selection as described in the last section.
   - Choose the desired command. As above, it doesn't matter where you find it. The most convenient option is to hold anywhere in the source selection; the context menu includes the relevant transfer verbs. Once a transfer verb has been selected, the Select command shortcut in the shortcut area is relabeled to indicate what transfer is in progress.
   - Hold on the destination spot to complete the transfer, as above.
3. Verb last (OneTouch Transfer).

- Make the source selection as described in the last section.
- Hold on the destination spot. Since no transfer verb has yet been chosen, all relevant verbs will appear on the context menu. Choose one of these to carry out the desired transfer.

As you can see, OneTouch Transfer is the most streamlined, especially if you know the relevant idioms: tap-swipe to select the source, hold on the destination till the menu appears, slide the pen/finger mouse/ to the desired verb, and release.

Any part of the display will work as a transfer target, provided the target isn't an object that refuses transfers. You can for example do a transfer into any of the margins, such as if you wanted to copy a command into the bookmark margin so that it's always handy when you're viewing that document (but only that document).

Text prompts in the lower margin guide the user when transfers are under way. For example, if you start by choosing the Copy command, the text would read "Use the pen or finger to select what to copy, then point at and hold the destination". If you start with the select command it would read "Use the pointer or finger to make your selection, then hold the flashing Select command for choices".

One problem that occurs on a touch device is that it's sometimes hard to point at the exact destination spot you intend, because the pen or fingertip gets in the way. To solve this, when you hold the destination, a transfer cursor appears along with the destination context menu. Before you choosing from the menu, you're free to use the pen or your finger to reposition the cursor (the menu will move out of the way as needed to let you freely move the cursor). When the cursor is exactly where you want it, you can proceed with choosing a menu action.

When you choose a verb to complete a transfer, like Copy Here, the transfer occurs, the source selection is cleared, and the PV exits select mode. The Select command shortcut stops flashing. If it's not configured as a persistent shortcut in the shortcut area it disappears.

Depending on where the destination point is, the destination context menu may include a few other choices not related to transfers. If the destination is in a link hotspot, then link action commands will appear on the menu (see Section 6.4.7); importantly, these include the Link Disable command by which you can turn the link off if you need to do a transfer that affects the anchor material. When a link is disabled, the anchor range acts just as if the link weren't present, although a muted emphasis of the link's anchor range remains so that you know a disabled link exists there.

If the command is not in a link hotspot, or the link is disabled, another command that may appear in the context menu is the name of the operation that would normally be performed when holding the destination spot, in absence of a pending transfer. For example, holding a note object normally opens it, so this action ("Open Note") will be present on the menu if the destination point is in the content area. That's important because it means you can do things like select text to copy, navigate inside a note, and then complete the copy there. Pending transfers never get in the way of your ability to navigate, and so multiple display windows for doing transfers are never necessary.

The Paste From Clipboard command is unlike other transfer commands in that it doesn't involve a source selection. Therefore, there are only two sequences for performing a paste:

- Verb first: Tap the Paste from Clipboard command, hold the destination, and choose Paste Here.
- Verb last: Hold the destination and choose Paste Here.

As you can see, verb first in this case is of value only to the naïve user, in that it fits into the model of picking commands from the Operating Manual, and it gives an extra little prompt for completing the paste ("Point at and hold the place you want to paste from the clipboard, or tap the Paste command again to cancel").

Cut, Delete, and Copy to Clipboard are analogous to Paste, in that they use a source selection only and no destination. Therefore, the two sequences that apply to these operations are

- Verb first: Tap the command, make the selection, hold in the selection (or on the command shortcut) for a context menu, and choose <verb> Now, e.g., Cut Now
- Verb last: Make the selection, hold in it (or hold the Select shortcut), and choose the verb. As for Paste, verb first is of value only to the naïve user.

6.5.2 Document Editing

Document editing includes all the traditional operations of inserting and deleting text and objects and manipulating text and object properties and styles. Editing also includes the undo/redo and save models of the PV. Since I don't intend this as a word processor spec, I will naturally only hit a few basic highlights of this set of operations.

6.5.2.1 Inserting and Deleting

As for everything else in the UI, inserting and deleting is supported by commands you can find in the Operating Manual, as well as in quicker to reach places like the Quick Sheet and context menus. There are also a variety of pen and keyboard shortcuts (gestures) for directly editing text. Input by pen of course requires text recognition. Otherwise you have to use the wireless keyboard.

Editing works with reference to an edit cursor. To place an edit cursor, tap within the line where you want it to appear. Any text you type on the keyboard is inserted at that point. To insert text using the pen, place an edit cursor and then use the insert gesture (a caret). Space opens up for you to write in, and widens as you fill it in. A short delay after you stop writing, excess blank space closes up, and the edit cursor is positioned just after the last character you wrote.

To delete text, you can select and delete as described in Section 6.5.1.2, or you can simply erase, using the erasing tip of the pen or the erase gesture with your finger. Note that erasing is an overwrite operation: blank space is left in place of what you erase. If you want to erase and close up the space, that's called delete; use the delete operation. A pen gesture for delete (the same one editors use for manuscripts) can be used as a shorthand for the select/delete process.

To insert an object at the edit cursor, hold for a context menu and choose the Insert... verb. You'll be offered a list of common objects you can insert here. This may not be everything you could possibly insert—it is a short list suggested by IntelliView. If more choices are possible, an Other Choices entry will exist that takes you to a full page (or pages) of choices laid out and explained in traditional web UI fashion. Clicking one of the choices inserts that kind of object and removes the choice pages from the navigation context.

You can also perform an insert without establishing an edit cursor first. Just hold to open a context menu and a tracking cursor will appear for you to adjust the insert point (see Section 6.5.2.1).

For every valid insert location there's a default insert choice that's shown on the context menu along with the general Insert... verb. For example, within a line of text the default provided is Insert Text; this gives you way to start inserting with the pen if you didn't already know the caret gesture. On the other hand, in between lines the default provides is Insert Note. In the bookmark margin, the default is Insert Bookmark. And so on.

6.5.2.2 Properties and Styles

You manipulate the properties of existing text and objects as you'd expect, via the Properties command, accessible via the Operating Manual, Quick Sheet, and context menus. Like the Cut and Delete commands, it works using a source selection, either verb first or last.

The property sheet that opens looks a lot like today's property sheets, except here the sheet is a document that's just like all other documents you view on a PV.

And because it's a document, it can take advantage of the full range of document structuring facilities that make it easy to read and find stuff in documents.

For example, the sheet has a table of contents enumerating each of its sections and an index that provides a flat list of all the properties. It also has tabs like today's property sheets, except here the tabs are just bookmarks in the property sheet document. You can search the property sheet if you want, and add your own notes to it. You can even copy commands from it to make customized sheets or to place individual property setting commands into the shortcut area (more on such customizations in Section 6.7).

The Properties command adds the property sheet to the navigation context as a new branch from the node containing the object you selected. Therefore, you can use the Next/Previous buttons to quickly switch between the document and the property sheet. On a dual-display PV, the page containing the selection shifts to the left display if necessary and the property sheet opens on the right, so you can see both at the same time. For some properties, as you change their values the results shows up in real time in the selection; an example is text styling properties. Other changes may not show up until you Apply or OK the properties.

As is the case for transfer commands, when you invoke the Properties command a Properties command shortcut appears in the shortcut area if one isn't already present. It blinks to indicate that property setting is in progress. You can hold the shortcut, or alternatively hold the source selection, to get context menu choices pertinent to the Properties command. Choices include OK Properties, Cancel Properties, and Apply Properties. Copies of these commands are also laid out on each property sheet page.

As you'd expect, OK closes the property sheet with the new settings in effect while Cancel closes it and reverts all property settings. Apply establishes the new settings and leaves the sheet open for more changes. The OK/Cancel verbs also remove the property sheet from the navigation context.

Individual property setting commands can exist independent of the sheet they're normally presented in. For example, if you copy the Bold command to the shortcut area or a toolbar, then you can embolden selected text just by tapping the Bold shortcut. (More precisely, when in the shortcut area or a toolbar the Bold command by default presents itself as a button that both toggles the bold attribute of the current selection and also controls the bold attribute of text added at the insert cursor.)

Styles work just like properties. Styles are collections of property settings and possibly template information and/or logic that an author or application developer has bundled together, named, and associated with a context the style applies to. You choose and manipulate styles the same as you do for properties. Styles that apply to larger sized objects like whole documents typically don't need you to select the whole object to invoke the style; the larger context can be inferred from the current selection or edit cursor.

6.5.2.3 Undo and Redo

The PV's undo/redo model builds on the concepts embodied in the navigation model: network structure, semantic clustering, and persistence (see Section 6.1.1). In fact, the undo/redo UI and features are almost identical to those provided by the Previous/Next commands, right down to offering an interactive graphical map of the undo/redo history and IntelliView choice lists.

Today's Office undo/redo model is basically that of a linear command chain, much as today's browsing model is a linear navigation history. In either case, you can undo/redo (previous/next) along the chain in a linear fashion, a keystroke or command at a time.

In contrast, the PV's undo/redo history, like its browsing history, accommodates branching of the command chain. For example, suppose you make a bunch of edits to a section of a document, then undo them all, then make some new changes, and finally decide you liked the original version better. With the network-structured command chain, you can undo all the latest changes to the section and then redo along the *prior* command chain, thus restoring your original edits. This command history can branch off in multiple directions and along multiple levels.

Now, it would be rather chaotic to try to navigate your way through such a rich command context if you could only see and choose from your alternatives at a keystroke/single-command granularity. This is like trying to make choices from a dinner menu using an electron microscope. So, the PV includes parsing knowledge that lets it cluster the command history, and therefore present it several levels of semantic detail.

For example, as you input and edit characters, the PV understand that you have just inserted or edited a word, a string of words, a sentence, a paragraph, and so on. At a higher level of grouping, it understands that you've just done a bunch of edits that constitute "the changes made to Section 2 of My Specification from 2:00 to 3:00pm on 12/31/97" as well as "all changes since last saving at 3:05pm on 12/31/97".

So, just as Previous/Next can offer choices of multiple places to jump to along the navigation chain, the Undo/Redo commands can offer choices of multiple points to jump to along the command chain. These may correspond to undoing the edits on the last word, sentence, paragraph, or section. By making appropriate choices, you can successively undo/redo at any of these levels of detail.

Similar to how audio clips are managed, the command chain is a database of commands indexed by time and by document, so it's possible to follow undos/redos across document boundaries, or only within the current document. Further, this database is persistent, meaning that you can undo all the way back to changes made in prior visitations to the document.

There may, of course, be implementation restrictions on the longevity of the command history. For example, a simple compromise in a first release would be to discard history each time a save is done, or better, to replace the history at each save point with a set of file diffs. With the former, you could not undo past the most recent save. With the latter, you could undo to any past save, though only at whole version granularity for anything prior to the most recent save.

The UI for undo/redo uses Undo and Redo commands, as you'd expect. If you're doing much editing, you'll probably want to put these in the shortcut area so they're always handy. They're also available via the Quick Sheet, an editing toolbar (Section 6.5.2.5), and the Operating Manual.

The Undo and Redo commands work exactly like Previous and Next. A single tap takes you one step back or forward along the command chain. A hold brings up a list of IntelliView guesses about degrees of undo/redo, based on recent undo/redo history and on the semantic knowledge of clustering.

The last choice on both the Undo and Redo menus is the undo/redo map. This works just like the navigational history and link preview maps, in that you can tap any node to undo/redo to that point, or hold the node to expand it's level of detail—for example, to delve from mapping section changes down to paragraph changes.

6.5.2.4 Creating and Saving

It should be pretty clear by now how you go about editing a document that already exists, but how do you get new documents into the PV? There is a New command for that. It works just like Insert (Section 6.5.2.1), except that the newly inserted object is added to the user's navigation context (as part of "my unfiled stuff") instead of being inserted into the current document.

Having created or edited a document, there comes the knotty question of saving.

The PV exploits its navigation context, persistence, and undo model to simplify the user's life when it comes to saving and filing documents. Basically, a user doesn't need to do either of these things, though they can when they want to.

In today's desktop world, we use the Save and Save As commands for a couple different, unrelated things. One is to explicitly persist changes, since things don't persist automatically. The other is to make filing choices, like deciding to file a copy of a document under a different name or in a different folder. The need to file things is also forced on the user by the save model, because the only way to persist a new document is to file it.

The PV simplifies matters by automating persistence, and by not requiring that things be filed in order to be persistent. Things that you don't explicitly file are simply held in system-managed file space that you can think of as, simply, "my unfiled stuff'. Even if you never file a thing, you can still always find it because it is part of your History Map, and is also included in some of the views on My Documents (for example, the view of all documents you've authored).

Not that filing things is a useless pursuit. There are certainly times when you want to control the physical location of an object, like when you want to copy it to a diskette or make it available in a shared folder for others to access. And some users just like to organize things into filing hierarchies as a way to keep track of them. Explicit filing also comes in handy when you want to say "Save this as My Spec v1.2 in My Documents", and then you go on making changes, so your working copy is now something later than v1.2.

As you'd expect, the PV provides Save and Save As commands to accommodate these filing needs. You don't *need* to use these, but you can, and to be honest, typically will.

In a perfect implementation Save would be a no-op, thanks to automatic persistence. However, it does serve a couple useful purposes. One concerns what happens when you start editing a document. If it's a shared document, you probably don't want your ongoing changes to be visible in realtime to other users, which they would be with automatic persistence. So unless there's a save semantic, you'll need to manually make a copy to edit rather than editing the original. A bit inconvenient.

The other useful purpose of explicit saves is to give a less than perfect implementation an opportunity to do needed housekeeping, like compressing the undo history (see last section).

So even though automatic persistence makes explicit saves theoretically unnecessary, the PV's save model does include this operation. It works like this. Any time you visit a document and start making changes, it's as if an implicit copy is made for you automatically. What you opened as "My Spec v1.2" becomes on the first edit "Working copy of My Spec v1.2". The original stays untouched, and all edits you make are persisted in the working copy. Tapping the Save command saves your changes back into the original and discards the working copy.

Let's dwell on this a second. Save is being used here not as a way to protect against data loss; that's automatic. It's being used as a way to create a snapshot that will be stable from ongoing changes; to file or publish a working copy.

Mainly you'd care to do this only when someone else might need to see the file in a useful state, or because it creates a convenient landmark for you in the undo history.

Still, from a user perspective, things seem pretty compatible with today's model: your changes don't appear in the original file until you save. To further smooth the transition, there's even a Close command. It prompts whether to save changes and removes the current document from the navigation context. For example, if you started editing "My Spec v 1.2" and closed without saving, you'd be back where you started—the original spec and no working copy floating around, same as today.

Now, under the hoods, something more efficient may happen than really making a working copy. For example, the original can be left intact and your edits accumulated as persistent deltas that only exist in your workspace, which only you see. These deltas constitute the undo/redo command chain described in the last section. When you save, the deltas are merged back into the original. Should the command chain become too long for efficient handling during an editing session, a real copy of the file can be forked off and updated at that point.

Note that activities like jotting, highlighting, and annotating don't do this "working copy" stuff, because by default those operations don't actually modify the underlying document. Those note taking functions work via external links. (If you *do* want notes carried internal to the document, you must set the notes source to be the document; see Section 6.4.8).

Save As, like Save, is also a still-needed operation, because it lets you fork off a version as a new file without updating the original. It works pretty much as you'd expect. Tapping the Save As commands opens a Save Sheet. This sheet has possibly more than one page, with a bookmark per page, providing access to the various applicable save options. Some of these pages may be document-type (app) specific.

The first page of the Save Sheet is the most common stuff—mainly, an edit field with the document name defaulted and a chooser for saying where to file it. The chooser includes a quick list of IntelliView guesses as to the most likely folders (typically a very short list). There's also a command shortcut that opens an explorer view on My Documents, from which you can navigate and select the destination folder if you need the full power of browsing or searching to find it. In that case, once you find the desired destination you select it using the normal selection idiom and OK the save. The OK/Cancel commands appear, as usual, in the context menus for the current selection and for the command shortcut (Save As in this case), as well as on the Save Sheet.

6.5.2.5 Editing Toolbars

For heavy duty editing there are enough applicable commands, properties, and styles that you want them very close at hand. The PV's web UI makes it fairly easy to find stuff a tap or two away, and to put a few shortcuts in the display margins, but that's not quite good enough for editing. Intensive editing just needs too many, and you want to be able to summon and dismiss them as a group. This is what toolbars are good for.

You've already read a bit about toolbars when I discussed the recorder controls in Section 6.4.6.2. The PV comes with a set of other toolbars like that one, but filled with sensible groupings of edit controls. For example, there are standard, formatting, margins/tabs, drawing, table, and other toolbars—the same kind of stuff Office has (although the presentation can be richer, since toolbars in the PV are HTML documents).

Whenever you start editing, the standard toolbar tab appears in the shortcut margin if it's not already present. Other editing toolbars may also appear, based on IntelliView heuristics. This decision is based on what kind of content you're editing, and on your own past toolbar usage patterns in this document and in general. You can dismiss any toolbar you don't want by erasing it's tab, and summon any not currently present by invoking the Toolbars command (available for example in the standard editing toolbar and on toolbar context menus). This command opens a categorized view on toolbars contained in the Operating Manual; "Editing Toolbars" is one of the categories and you can expand that and make selections therein (toolbars already on the display are shown already selected).

You can use any or all of the editing toolbars, in any combination, just as in Office. The main difference is that a PV toolbar can hide itself under a toolbar tab in the margin; you open and close it by tapping the tab. You can also make a toolbar float by holding its tab and picking that option from the context menu (or by setting it on the toolbar properties).

When you have multiple, non-floating toolbars open they tile (stack) themselves along the edge their tab lives on, so you can create the same kind of visual arrangement of toolbars as Office has. You can also nest toolbars, which provides a space efficient way to make related sets of functions handy. What you see in this case is a toolbar that contains toolbar tabs. Clicking a nested tab opens the nested toolbar into its parent's frame.

Because toolbars are just documents, they're easy to create and customize. This is explained in Section 6.7.

6.6 Built-In Documents

Under Construction.

- Sign-in Page
- Start Page
  - Personal newsletter
  - Desktop access (docs, folders, mail, PIM, web)
- Map book
  - Site, history, topic (browsing), net neighborhood, vicinity, local machine
- Operating Manual
  - Help
  - Quick Sheet
    - UI Shortcuts (incl docs and people)
  - Settings Sheet
  - Status Sheet (logon, network, and PV status)
  - Context menus (HTML authorable)
  - Toolbars
- My Documents
  - All
  - Favorites (HTML authorable)
  - Stored here
  - Stored elsewhere in my workspace
  - Authored by me
  - Recent
  - Oldies
  - Subscribed to
  - Purchased
  - Read
  - Unread
  - Explorer
- Annotations Folder
  - Clippings Folder
  - Notes Folder
  - Bookmarks Folder
- Notebook
- Address book
- Mailbox
- Calendar

6.7 Customizing

Under Construction.

- Settings
- Customizing the quick sheet
- Customizing UI idioms
- Customizing command and document shortcut margins
- Customizing individual commands (changing their text/appearance)
- Pulling commands off prop sheets
- Customizing prop sheets and context menus (they are just html)
- Creating and customizing tool bars

Putting stuff in the margins: how the diff margins behave

6.8 Desktop Sync

Under Construction.

Focus tracking
Workspace sync
IR routing via desktop

6.9 Collaboration

Under Construction.

Mail
PIM
Meetings / collaboration
  IntelliView features in a meeting
  Presentation Sync
  Private presentation slides and note taking
  Chat and instant messages
  Shared workspace
  Controlled meetings
Internet phone

6.10 Application to Desktop UI

Under Construction.

Scroll-less, windowless, widgetless, desktopless
  Document orientation
  Page display (scroll-less)
  Invisible Web UI w/ shortcuts
  Link and command models
  Network navigation context / clustering
  Persistence model
  Multiple browsing instances
Sign in
IntelliView
Built-in docs
Universal reading and note taking activities
IntelliTape recorder
Editing, undo, and save models
Customization
Collaboration
Application model

7 Content and Content Services

Under Construction.

Online / print formatted file format
  Presentation information (page size, orientation, scaling ranges)
  Multiple layouts in one file
  Netdocs
  Acrobat, Postscript
  HP PCL, other print formats
Books
Periodicals
Streaming media
Content and PV size formats
Limited and unlimited use
Subscriptions and back issues
Authorization and revocation
Content crypto / right to copy
Business opportunities

8 Software and Protocols

Under Construction.

Win CE
ZAW
Sign in
Nearby machine mapping
Focus tracking
Desktop routing
App / shell interfaces
Mail and PIM
Streaming media
Push delivery
Content purchasing and subscription

9 Display Technologies

Under Construction.

Reflective LCD (Sharp Super Mobile LCD)
Dynamic Excite Drive (Bright Lab Co.)
Flexible LEP (Cambridge Display Technology)
Foldable OLED on metal foil (Princeton)
Polysilicon TFT's on plastic (Livermore)
FEDs (Mircron, PixTech, FED)

10 Personal Viewers and Other Handhelds

Under Construction.

PDAs

Wallets
Wireless phones